FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

DEGREE PROGRAMME IN ELECTRONICS AND COMMUNICATIONS ENGINEERING

MASTER'S THESIS

COMPARISON OF MULTI-LAYER BUS INTERCONNECTION AND A NETWORK ON CHIP SOLUTION

Author	Niko Huttula
Supervisor	Jukka Lahti
Second Examiner	Jussi Jansson
Technical Advisor	Antti-Kalle Länsman

March 2019

Huttula N. (2019) Comparison of Multi-Layer Bus Interconnection and a Network on Chip Solution. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Electronics and Communications Engineering. Master's Thesis, 47 p.

ABSTRACT

This thesis explains the basic subjects that are required to take in consideration when designing a network on chip solutions in the semiconductor world. For example, general topologies such as mesh, torus, octagon and fat tree are explained. In addition, discussion related to network interfaces, switches, arbitration, flow control, routing, error avoidance and error handling are provided. Furthermore, there is discussion related to design flow, a computer aided designing tools and a few comprehensive researches. However, several networks are designed for the minimum latency, although there are also versions which trade performance for decreased bus widths. These designed networks are compared with a corresponding multi-layer bus interconnection and both synthesis and register transfer level simulations are run. For example, results from throughput, latency, logic area and power consumptions are gathered and compared.

It was discovered that overall throughput was well balanced with the network on chip solutions, although its maximum throughput was limited by protocol conversions. For example, the multi-layer bus interconnection was capable of providing a few times smaller latencies and higher throughputs when only a single interface was injected at the time. However, with parallel traffic and high-performance requirements a network on chip solution provided better results, even though the difference decreased when performance requirements were lower. Furthermore, it was discovered that the network on chip solutions required approximately 3-4 times higher total cell area than the multi-layer bus interconnection and that resources were mainly located at network interfaces and switches. In addition, power consumption was approximately 2-3 times higher and was mostly caused by dynamic consumption.

Key words: NoC, Interconnect, CAD tool, ASIC, Topology, Performance, Area and power consumption.

Huttula N. (2019) Monitasoisen väyläarkkitehtuurin ja tietokoneverkkomaisen ratkaisun vertailua. Oulun yliopisto, tieto- ja sähkötekniikan tiedekunta, elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Diplomityö, 47 s.

TIIVISTELMÄ

Tutkielmassa käsitellään tärkeimpiä aihealueita, jotka tulee huomioida suunniteltaessa tietokoneverkkomaisia väyläratkaisuja puolijohdemaailmassa. Esimerkiksi yleiset rakenteet, kuten verkko-, torus-, kahdeksankulmio- ja puutopologiat käsitellään lyhyesti. Lisäksi alustetaan verkon liitäntäkohdat, kytkimet, vuorottelu, vuon hallinta, reititys, virheiden välttely ja -käsittely. Lopuksi kerrotaan suunnitteluvuon oleellisimmat välivaiheet ja niihin soveltuvia kaupallisia työkaluja, sekä käsitellään lyhyesti muutaman aiemman julkaisun tuloksia. Tutkielmassa käytetään suunnittelutyökalua muutaman tietokoneverkkomaisen ratkaisun toteutukseen ja tavoitteena on saavuttaa pienin mahdollinen latenssi. Toisaalta myös hieman suuremman latenssin versioita suunnitellaan, mutta pienemmillä väylänleveyksillä. Lisäksi suunniteltuja tietokoneverkkomaisia ratkaisuja vertaillaan perinteisempään monitasoiseen väyläarkkitehtuuriin. Esimerkiksi synteesi- ja simulaatiotuloksia, kuten logiikan vaatimaa pinta-alaa, tehonkulutusta, latenssia ja suorituskykyä, vertaillaan keskenään.

Tutkielmassa selvisi, että suunnittelutyökalulla toteutetut tietokoneverkkomaiset ratkaisut mahdollistivat tasaisemman suorituskyvyn, joskin niiden suurin saavutettu suorituskyky ja pienin latenssi määräytyivät protokollan käännöksen aiheuttamasta viiveestä. Tutkielmassa havaittiin, että perinteisemmillä menetelmillä saavutettiin noin kaksi kertaa suurempi suorituskyky ja pienempi latenssi, kun verkossa ei ollut muuta liikennettä. Rinnakkaisen liikenteen lisääntyessä tietokoneverkkomainen ratkaisu tarjosi keskimäärin paremman suorituskyvyn, kun sille asetetut tehokkuusvaateet olivat suuret, mutta suorituskykyvaatimuksien laskiessa erot kapenivat. Lisäksi huomattiin, että tietokoneverkkomaisten ratkaisujen käyttämä pinta-ala oli noin 3-4 kertaa suurempi kuin monitasoisella väyläarkkitehtuurilla ja että resurssit sijaitsivat enimmäkseen verkon liittymäkohdissa ja kytkimissä. Lisäksi tehonkulutuksen huomattiin olevan noin 2-3 kertaa suurempi, joskin sen havaittiin koostuvan pääosin dynaamisesta kulutuksesta.

Avainsanat: NoC, Yhteys, CAD-työkalu, ASIC, Topologia, Suorituskyky, Alue- ja tehonkulutus.

TABLE OF CONTENTS

FOREWORD

An objective of this research was to discover differences between network on chip solutions and commonly used multi-layer bus interconnection. For example, how much larger would be the area and power consumptions be and what kind of performance differences can be expected. In addition, it was important to discover the main reasons for such results and how much those can be affected by design choices. The thesis was produced at Nordic Semiconductor Finland from May 2018 to March 2019.

At first, I would like to thank my manager Pekka Kotila for this magnificent opportunity to work at Nordic. I would also like to thank my technical advisor Senior R&D Engineer Antti-Kalle Länsman for his guidance and support during the work flow. In addition, many thanks to Senior System Architect Hannu Talvitie for his contribution and for providing the thesis subject. Furthermore, I would like to thank University Lecturer Jukka Lahti for supervising the thesis and for his great work at the University of Oulu. Also, thanks to Docent Jussi Jansson for being the second examiner. Great many thanks to all my colleagues for the support.

Oulu, March 12, 2019

Niko Huttula

LIST OF ABBREVIATIONS AND SYMBOLS

ABR	Available Bit Rate
ACE-Lite	AXI Coherency Extensions Lite
ACK/NACK	Acknowledge/Not Acknowledge
ADB	AMBA Domain Bridge
ADS	Advanced Design System
AHB	Advanced High-Performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
AXI	Advanced Extensible Interface
BE	Best Effort
BFT	Butterfly Fat Tree
BVCI	Basic Virtual Component Interface
CAD	Computer-Aided design
CBR	Constant Bit Rate
CCI	Cache Coherent Interconnect
CCN	Cache Coherent Network
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DC	Design Compiler
DEF	Design Exchange Format
DFT	Design-For-Test
DMA	Direct Memory Access
DPA	Dynamic Priority Arbiter
DRAM	Dynamic Random-Access Memory
DSP	Digital Signal Processor
ECO	Engineering Change Order
EMI	Electromagnetic Interference
FCFS	First Come First Served
FIFO	First In First Out
FPA	Fixed Priority Arbitration
FPGA	Field Programmable Array
GUI	Graphical User Interface
HDL	High-level Description Language
HVT	High Threshold Voltage
IC	Integrated Circuit
IP	Intellectual Property
ISO	International Standard Organization
I2C	Inter-Integrated Circuit
LPD	Low-Power Distributor

LVT	Low Threshold Voltage
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MWD	Multi-Window Display
NI	Network Interface
NIC	Network Interconnect
NIU	Network Interface Unit
NoC	Network on Chip
NS2	Network Simulator 2
OCP	Open Core Protocol
OSI	Open System Interconnection
OVM	Open Verification Methodology
PIF	Processor Interface
PIP	Picture-In-Picture
QoR	Quality of Results
QoS	Quality of Service
QVN	QoS using Virtual Network
RTL	Register Transfer Level
SAF	Store and Forward
SDC	Synopsys Design Constraints
SoC	System on Chip
SPI	Serial Peripheral Interface
SPIN	Programmable Integrated Network
SSP	Synchronous Serial Port
SVT	Standard Threshold Voltage
TCP/IP	Transmission Control Protocol / Internet Protocol
TLX	Thin Links
UART	Universal Asynchronous Receiver-Transmitter
UPF	Unified Power Format
USB	Universal Serial Bus
UVM	Universal Verification Methodology
VBR-NRT	Variable Bit Rate - Non-Real Time
VBR-RT	Variable Bit Rate - Real Time
VCT	Virtual Cut Through
VIP	Verification IP
VLSI	Very-Large-Scale Integration
VMM	Verification Methodology Manual
VOPD	Video Object Plane Decoder
WH	Wormhole
XHB	AXI-to-AHB Bridge

C_L	Load capacitance
C_T	Parasitic capacitance
V_{dd}	Supply voltage

f_{clk}	Clock frequency
V_T	Threshold voltage
R_T	Parasitic resistance
L_T	Parasitic inductance
a	Activity factor
β	Gain factor
τ	Rise and fall time

1. INTRODUCTION

Network on Chip (NoC) solutions have become more researched topic for last 20 years and its basic idea can be compared with worldwide interconnect that is commonly known as Internet. However, the protocols used are simplified since not all the features are required and small proximity between used logic units allows less complex transaction methods. For example, whereas Internet uses Transmission Control Protocol / Internet Protocol (TCP/IP) based transactions, NoC may use simpler and more efficient solutions such as circuit switching. However, the most important requirements of NoC are to provide low latency and high performance with reasonable Quality of Service (QoS), area and power consumption, although the final solution is more often a tradeoff between them. Furthermore, NoC solutions have also grown in popularity due to the reason with smaller processing technology there is higher delay with wirings than with logic gates. The propagation delay of wires causes issues when it exceeds the clock period which in the worst case causes the underuse of the technology and back-end timing difficulties. Thus, the efficient implementation is somewhat problematic with common networks due to their poor scalability and when the design complexity increases even more, the interconnection becomes a bottleneck. [1]

Topologies of NoCs may vary between commonly known networks such as mesh, torus, ring, fat tree, butterfly and scalable programmable integrated network (SPIN) but their combinations are also possible. The basic idea is to share resources between transactions either in time or space which leads to the better utilization of networks, but it may cause issues such as congestion, deadlock, livelock and starvation. To minimize such situations there are routing, flow control and error handling that must be considered. For example, a solution could be to design an error resilient NoC or use a deterministic routing method which guides traffic based on the current load of the network. In addition, accurate designing at the beginning of register transfer level (RTL) phase helps to minimize problems and the workload on later design phases. For example, with such designing it is possible to find errors which might not be found until the gate-level phase due to the complexity of design. Furthermore, the specific structure of NoCs can be designed with several Computer-aided Design (CAD) tools. For example, there are design tools for simulating the chosen topology at higher level, to implement required components inside the network and to map these components to each other. In addition, there are also complete toolsets which generate and simulate NoCs from the given parameters. [1]

The aim of this study was to discover how would NoC perform when compared with multi-layer bus interconnection and to figure this out, a reference network was built and its functionality was modeled for several NoC solutions. For example, several requirements such as interface protocols, interconnection maps, arbitration and minimum timing and throughput requirements were pre-specified. However, the generic structure of NoCs was left for used tool to generate. Furthermore, the difference between overall performance, area and power consumption will be presented and a conclusion whether the NoC is worth using over the reference multi-layer bus interconnection will be discussed. For example, it was found out that the NoC had a few times higher area and power consumption, but the overall performance was better. However, results were not as uniform as that since the outcome depended on used test pattern and performance requirements.

2. NETWORK ON CHIP BUS ARCHITECTURES

An increase in design complexity usually leads to larger bus lengths which with the larger node count has a huge impact on the efficiency of different architectures. However, the architecture and used protocols are usually determined by the common goals such as maximum latency, minimum throughput, area and power consumption, scalability and reliability. Nonetheless, different topologies cause tradeoff between these goals which may cause power or latency to grow over the given limits. [1]

2.1. System on Chip

System-on-chip (SoC) can be defined as an Integrated Circuit (IC) which consists of multiple stand-alone very-large-scale integrations (VLSI) which together models full functionality for a specific application. For example, SoC may contain building blocks such as microprocessors, memories, digital signal processors (DSP), audio, video and graphic controllers. An example of a SoC structure can be seen in Figure 1 [2 p. 7] where used building blocks may be created by the designers themselves or be bought from the vendors. However, these blocks are usually represented in high-level description language (HDL) or as an optimized transistor-level layout. Furthermore, these blocks can also be determined as soft, firm or hard cores. Soft core refers to blocks which are reusable and synthesizable at RTL, but firm cores are provided as synthesized code or as a netlist and are usually optimized by structurally and topologically. The latter leads to better performance and smaller logic area for the specific functionality. However, hard core blocks may only exist in strict layouts or as fully placed and routed netlists which are designed for the specific process technology and are the most optimized for performance, power and area. [2 p. 3-8]

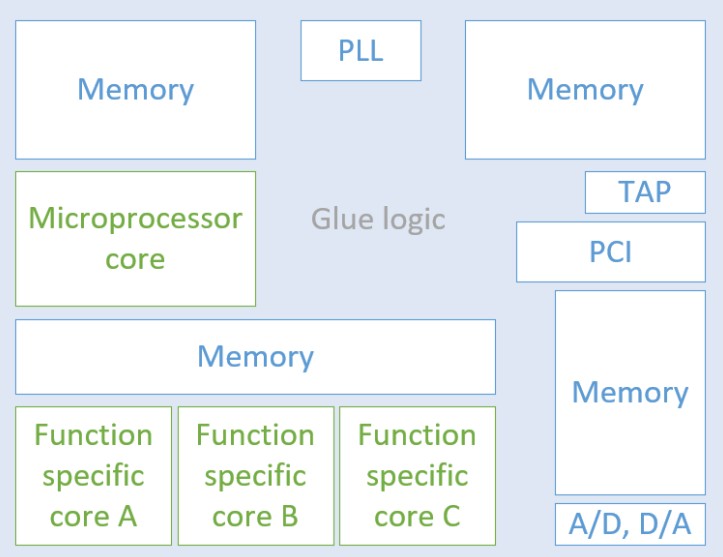

Figure 1. An example of system-on-chip structure.

Issues usually occur due to the complexity of design and are related to interfaces, synchronization, data management, design verification, testing and architectural and system-level choices. For example, there may be difficulties with core-to-core communication, although there are several architectures such as IBM's processor local bus [3], ST Microelectronics STBUS [4] and ARM's advanced microcontroller bus (AMBA) [5]. However, these architectures tend to be designed for the specific processors which may cause difficulties when utilizing different building blocks. [2 p. 8, p. 113-115] Figure 2 shows shared-medium bus architecture which has been commonly used in SoCs [1 p. 25]. Such architecture has issues with the arbitration which causes an increased idle time and thus the performance may decrease, even though the shared-medium bus architecture benefits from the asymmetric communication when only a few masters interact with many slaves. However, due to limited scalability its performance efficiency decreases highly with larger networks which leads to a bottleneck effect. In addition, shared-medium bus architecture has low energy efficiency due to high switching capacitance and functional congestion. [1 p. 25-27]

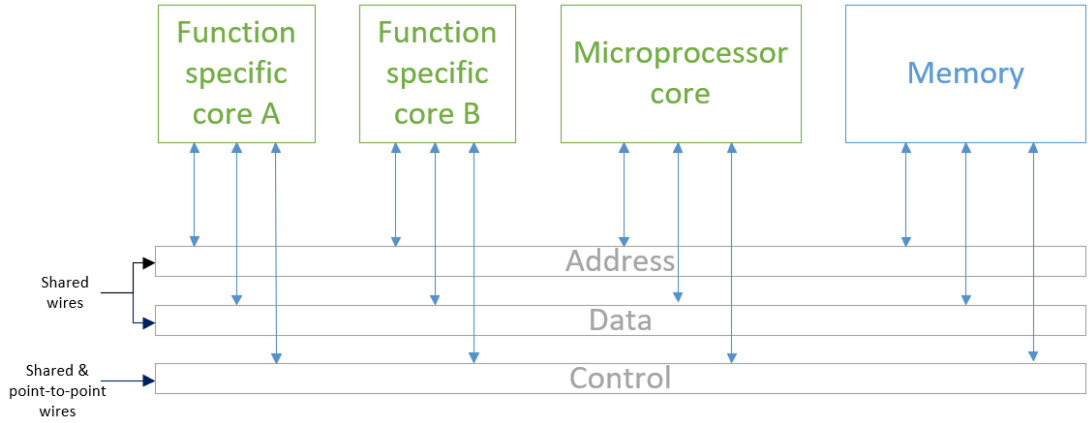

Figure 2. Commonly used shared-medium bus architecture.

The propagation delay of wires begins to limit transaction speed when the size of process technology decreases which causes also a bottleneck effect. However, the higher utilization of upper metal layers can still provide high-bandwidth between Intellectual Properties (IP). [1 p. 1, 6 p. 3] In addition, more difficulties are caused by the frequency increase which makes impedance characteristics be less precise which causes increased impedance mismatch. Also, increased capacitance and the skin effect of wires, i.e. higher resistance, are causing issues which are growing even larger with the situation where more logic is implemented in the same space as before. [1 p.47] Furthermore, the error probability increases due to lowered voltage levels which cause noise margins to be smaller and a chance for electromagnetic interference (EMI) to be higher. On the other hand, crosstalk is more likely to occur due to the difficulty of detecting all on-chip noise sources. In addition, the probability of synchronization failures and metastability increases due to timing noise, transmission speed and clock domain changes. Also, spurious pulses may have an influence on signals. [1 p. 3-5]

2.2. Network on Chip

For NoC solutions there are several main topics to be discussed. For example, what topologies are used, what kind of components are implemented inside networks, what is the required data rate, how data flow is controlled and how congestion and errors are avoided and handled. Furthermore, there are several CAD tools to aid with higher level simulation, component mapping and RTL generation.

2.2.1. Basic topologies

Different topologies match better with different NoC objectives which can be application specific, reconfigurable or general networks. These networks can also be an irregular type where they are optimized for the specific use of applications which can be achieved by removing unnecessary data paths and routers to gain a lower cost and still have enough performance. For example, such networks can be customized to avoid problems like hotspots and networks on Application Specific Integrated Circuits (ASIC) usually are irregular types but there can be single or hybrid versions where shared medium, indirect and direct networks are used. Within the direct network each node has a network interface (NI) block, which is also called a router, that is then again connected directly to other neighbor routers. Overall, these nodes are on-chip computational units which use point-to-point channels to communicate with each other. Increasing the number of nodes also grows the total bandwidth of networks which is the reason why direct routing is a popular choice to be used in large-scale, although the down side is higher area and power consumption. However, within the indirect networks, communication between different NIs is guided through switches which are being connected each other so that a node could communicate with other nodes. The fundamental difference between direct and indirect networks is that indirect network provides a programmable connection without any data processing NIs which leads to a simpler data delivery. [1 p. 24-34, p. 159]

Crossbar topology is simple but not the very scalable version of network, where all nodes are connected to each other or similarly where processing elements are connected to each other via single switch. Crossbar topology is represented in Figure 3 a). [1 p. 28-30, p. 158] However, mesh topology is a simple version of network where nodes are placed in two-dimensionally and are connected to their neighbors which can be seen in Figure 3 b). The number of nodes grows logic area linearly and the distance between nodes may be long which also gives its addition to power consumption. Performance of mesh topology suffers under heavy load, but it can be reduced by adding bypass links. However, this means that care must be taken with transactions to gain full benefit of the topology. [1 p. 34] In addition, there is also a possibility to use n-dimensional cube structure such as a two-dimensional hypercube that can be seen in Figure 3 c). [1 p. 28-30]

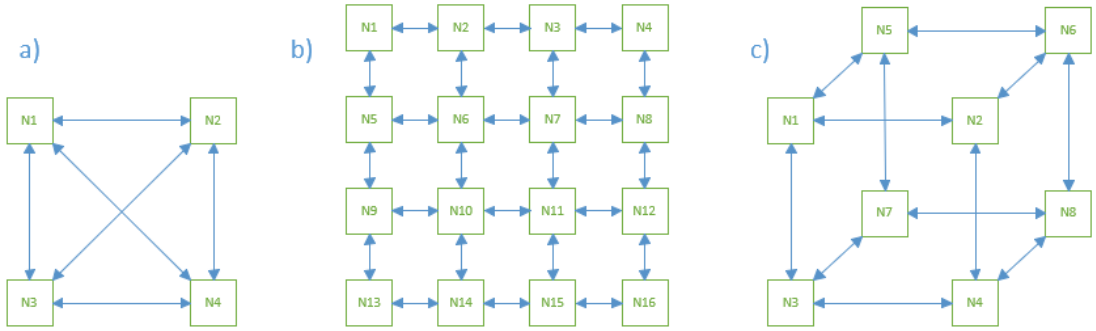

Figure 3. Direct network topologies are a) crossbar, b) two-dimensional mesh and c) two-dimensional hypercube.

Such as mesh, torus-based topology is also a simple version of network where nodes are placed on a ring or torus as it can be seen in Figure 4 a) [1 p. 30]. With torus topology, the area and power consumption increase linearly with the number of nodes and their distance to each other. In addition, its performance decreases when the ring size increases due to the shared band, even though the performance can be increased by adding more dimensions and decreasing link lengths. All in all, the overall power consumption and performance are better with torus than with mesh topology, although their logic area is quite much the same. [1 p. 34]

Octagon is a topology where every node is connected to other so that the maximum movements between the nodes require at the maximum of two jumps which can be seen in Figure 4 b) [1 p. 36]. Also, connecting other octagon networks to each other is a possible solution. Furthermore, there is a polygon structure where connection possibilities are increased by the structure which reminds a spiderweb and is derived from the octagon topology. [1 p. 35-36]

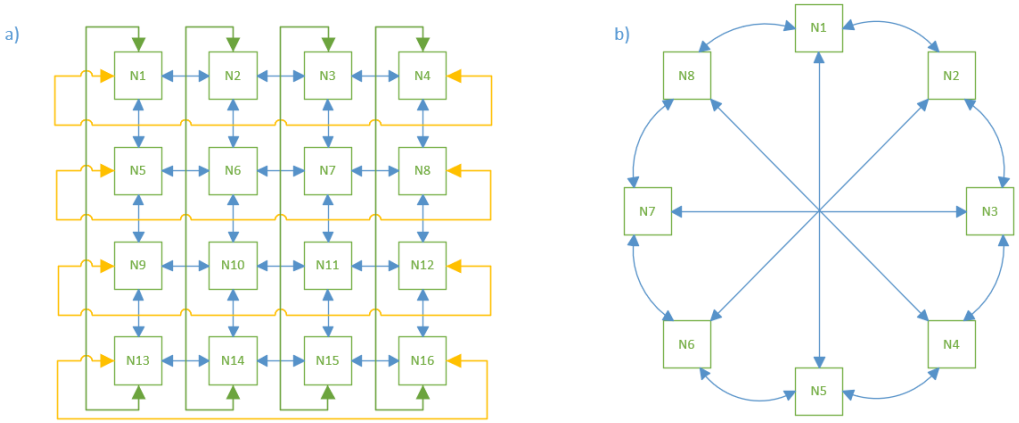

Figure 4. Direct network topologies are a) torus and b) octagon.

Figure 5 a) [7] shows a fat tree topology where a switch is connected to switches below and from them again to sub switches or nodes which causes network to be indirect. However, horizontal moving to other switches of different paths of the tree is not possible. The structure is simple and effective, but the paths are fixed which increases the chance of shared bandwidth. Furthermore, the number of required switches is relatively high when it is compared with nodes which makes it more difficult to be implemented in the larger scale. However, the bandwidth issue can be

reduced by duplicating the paths of the tree, but the downside is that also required area increases and with larger designs the layout becomes more complex and difficult to implement when it is compared with mesh and torus topologies. [1 p. 32-35] A butterfly topology is achieved by adding nodes on both sides of the tree which doubles the node count but does not grow the switch count. The butterfly topology can be seen in Figure 5 b) [7]. However, the down side of such topology is that it uses only deterministic routing and thus it does not have diversity in paths and has longer wirings. In addition, Figure 5 c) shows a SPIN topology that is one of the earliest NoC and which is meant for packet switching, although the topology itself is derived from the fat tree. [1 p. 35, 7]

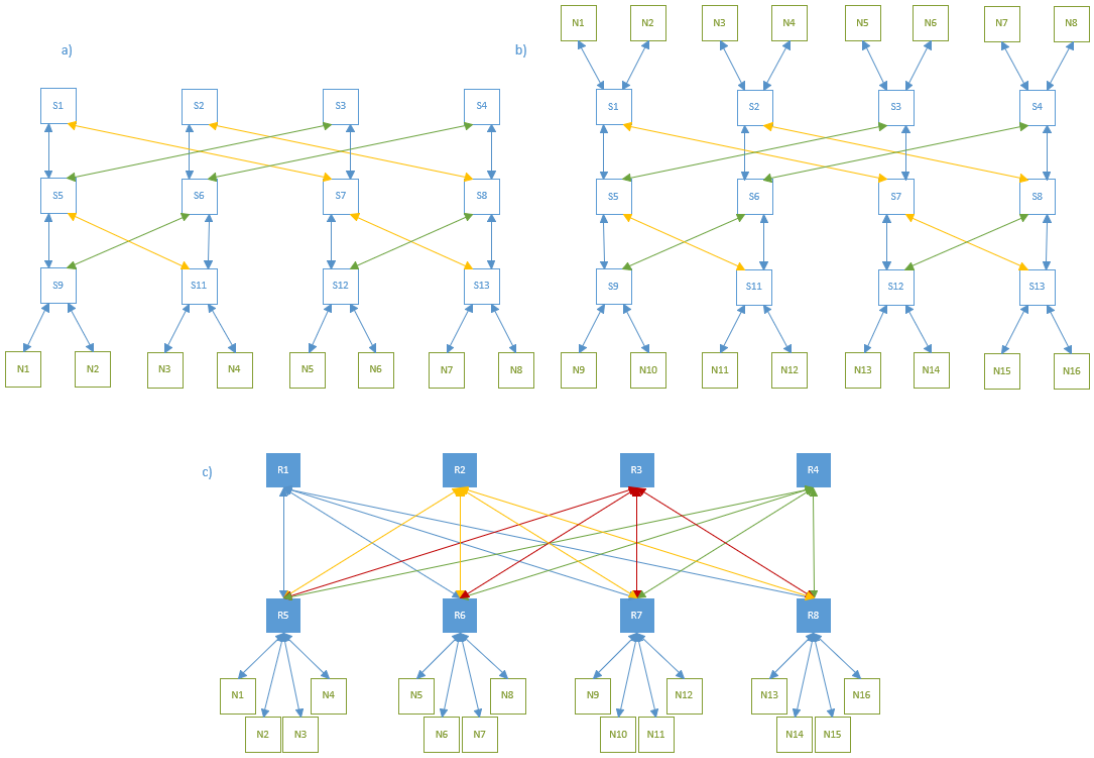

Figure 5. Indirect network topologies are a) fat tree, b) butterfly and c) SPIN.

2.2.2. Building components of network

Switches are used to route data from the input to the desired output and the structure of the switch may contain input and output buffers, an interconnection matrix and a control circuit. An example structure of the switch can be seen in Figure 6 a) [6]. However, when the buffers are used inside the switch, the power consumption increases significantly which leads to the designs such as mesh architecture where buffers are avoided. The interconnection structure of the switch can be thought as a multiplexer when the logic level is considered but it can also be implemented with a single crossbar or as cascaded stages. In addition, to control the specific use of the switch, there is an external controlling circuit required which can also take care of arbitration, part of the flow control and error detection and correction. Furthermore, virtual channels can be used to avoid deadlocks and to maximize the usage of channels.

For example, a switch, which contains first in first out (FIFO) buffers, can be divided into virtual channels that can be seen in Figure 6 b) and c) where each input and output multiplexer have two virtual buffers. Physically, these virtual channels can be simply one FIFO buffer which has a complex control protocol to select correct messages for each destination. Such structure reduces the number of blocked transactions, e.g. deadlocks, when the parallel transactions are possible. In addition, virtual channels improve latency and throughput when the messages can wait inside virtual buffers, instead of being resend afterwards. [1 p. 38, p. 160-163]

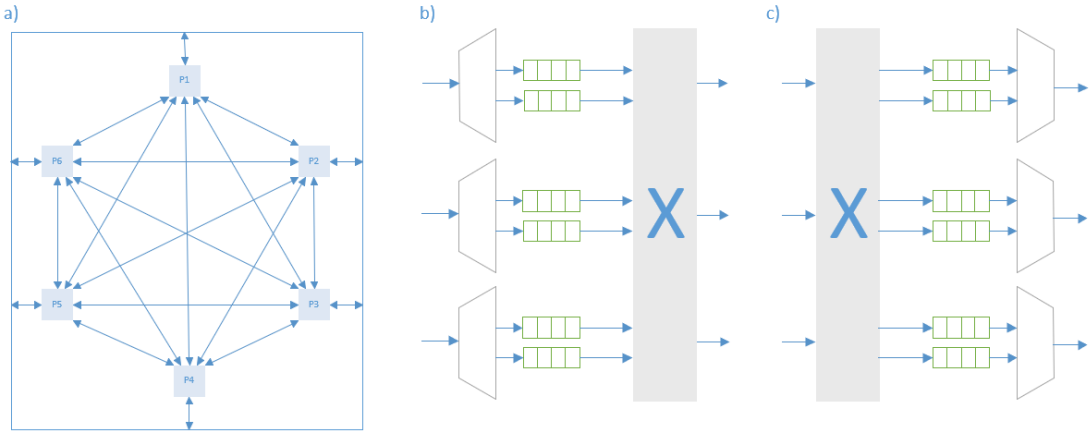

Figure 6. a) is an example interconnection within switch and b) and c) are an example view of virtual channels within switches.

Switches can be divided into packet and circuit switches. Whereas simple circuit switch opens a fixed data path to send all its data to the destination, a packet switch sends multibit data packets via various paths. However, the choice to use a specific switch type affects achieved QoS, cost and complexity. For example, circuit switches remind a SoC bus where the transmission latency is low but the latency to initiate the desired path is higher. In addition, required area in form of routers is lower with circuit switches but those do not tend to scale well due to reason that the formed link reserves resources from the other switches. For these reasons packet switches are usually preferred in large NoC, although the down side of packet switches is that delay varies between transacted packets which may cause long waiting times before the whole data is transmitted, which causes QoS to be harder. Furthermore, transaction delay varies between different packet switching schemes which are store and forward (SAF), virtual cut through (VCT) or wormhole (WH) switching. SAF is the slowest and most basic type of scheme. WH is taken somewhat further from VCT where all the packets are sent in a row without added waiting time. WH transmits a packet whenever the receiver has free space and it is most commonly used with NoCs. [1 p. 38, p. 160-169]

Network Interface (NI) is a bridge between transport and transaction layer that is implemented to offer a protocol view for the external logic and to convert external protocols to an internal protocol of NoC. The functional idea can be seen in Figure 7 [1 p. 210] which shows a generic structure of NIs and where an open core protocol (OCP) [8] is converted to the internal protocol. The structure reminds a session layer in the International Standard Organization / Open System Interconnection (ISO/OSI) model which functionality is to determine when the session is opened, time used and when to be closed. In addition, NIs also control transmission during the open stage, supports flow control, security, QoS and provides routing tables for external IPs to

locate each other. The benefit of using NIs is that the protocol inside NoC does not need to be processor core specific but it can rather be freely chosen which eases core designing when there is no need for advanced knowledge of other end systems. Furthermore, NIs can be divided into masters and slaves which use commands such as writes and reads. Masters initiate transactions with requests which are commands and data transactions that slaves receive and behave accordingly. In addition, there may be a response phase involved where slave sends request to its master. For example, there can be an acknowledgement bit in each transaction or a return data request. NIs should also be able to communicate with each other so that they do not affect unwanted NIs between the request and response phase, otherwise, deadlocks might occur. However, deadlock free transactions can be achieved when the isolation of connections is formed either in space or time. The space isolation would require separated resources and the time isolation would require division into time slots which gives turns to use resources. [1 p. 39, p. 210-212, 6 p. 6-7]

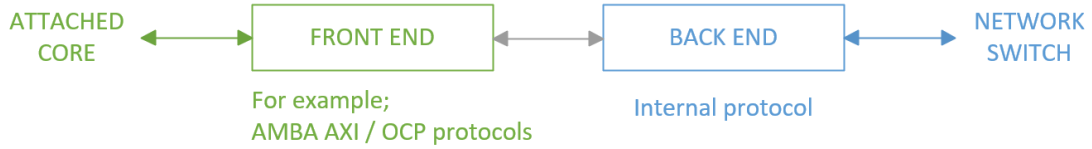

Figure 7. A generic structure of Network Interface.

Interconnection between switches and nodes is usually made with physical wires because of its relatively low cost and metal layer availability. This leads to design where data and control are separated which causes design to be less complex and performance to increase. However, the capacitance and resistance of wires grow larger based on its length which increases the propagation delay of the path quadratically. Longer wires have also more dispersion and voltage drop is larger which with the delay may cause problems, although issues can be compensated by the pipeline registers and repeaters. [1 p. 38-39] Furthermore, NoC structure which uses switch-based networks does require addresses and identifiers to connect NIs correctly. It is common to use logical addresses instead of physical ones to hide SoC structure or to share common address space. When logical addresses are used it does usually require remapping which can be done on software or hardware level. For example, it can be done on run time or during the designing phase as centralized at one location or as distributed over NoC. In addition, there might also be requirements for different levels of QoS and transaction reordering. In these situations, identifiers can be used. [1 p. 168-169]

Every node and switch can involve arbiter and multiplexer pair which must be optimized as required. An important job with the arbiter is to allocate resources fairly which can be done with fixed or dynamic arbitration. Dynamic Priority Arbiter (DPA) has logic which decides which requests to grant access and which to promote based on its current state. There are arbiters like round-robin which grants access equally in a loop and more complex types such as first come first served (FCFS) which has its own ordering profile. Fixed Priority Arbitration (FPA) is based on its input signal order which determines the outcome and does not need other priority states. [6 p. 61-68]

Designing ASIC and field programmable array (FPGA) networks differ from each other in a way that ASIC does not necessarily need to be configurable on the run. There is usually an exact need for specific functionality with ASICs where network parameters, layout, link capacity allocation, buffer sizes, packet headers, partial routing tables and services are well known. Leading to the point, where all the later

configurations are not needed and can be left out which simplifies the design and decreases the area and power consumption. However, FPGAs are more computational flexible than ASICs which makes them more viable to run different types of applications. [1 p. 150-152]

2.2.3. Network traffic and routing

NoCs have QoS requirements with module-to-module traffic, data rates, statistical behavior and predictability that should be considered when the correct network architecture is being selected. In addition, variables like signal loss, delay, priorities and actions should be considered. Ultimately, this leads to design switching techniques, topologies, addressing and routing schemes which are used inside NoC. Furthermore, there is a need for end-to-end mechanism which provides reliable delivery, a connection of the modules, flow control, management of receiver buffers and control access to multiple resources. Also, a network-level congestion control and soft error handling should be taken on account to handle extreme conditions, excessive traffic and data corruption. However, because modules outside the NoC often behave in various ways, the NoC should support multiple QoS requirements. [1 p. 147-148]

The reliable traffic of NoCs can be derived from the asynchronous transfer mode (ATM) and QoS service classes such as IntServ and DiffServ which are Internet protocol-based network standards. Whereas IntServ requires complex implementation, DiffServ does remind ATM traffic but it has five different service classes. However, these classes cannot be directly mapped to NoCs but rather with modifications. ATM is consisted of constant bit rate (CBR), variable bit rate – real time (VBR-RT), variable bit rate non-real time (VBR-NRT), available bit rate (ABR) and best effort (BE). When resources can be reserved before the transmission and the low delay and loss are required, CBR and VBR techniques can be used. ABR technique can be used when the band is allocated with several routers. BE offers sort of filler technique which can be used to utilize the rest of the band whenever it is possible. Furthermore, circuit switches usually use CBR or VBR technique because its transmission delay and loss are low and fixed. Packet switches, on the other hand, do more often use BE technique because they do not fit well with CBR or VBR techniques due to the high initiation delay of channels. [1 p. 154-158]

NoCs usually trade between different objectives such as power, area and VLSI resources, performance and robustness to traffic changes. Power consumption is usually wanted to be as low as possible which can be achieved when traffic is routed via the shortest path or when every router and link are separately optimized. Area and VLSI resource usage can be minimized with the different routing mechanism, although the lack of resources affects performance in a sense of increased delay. For example, changes can be made with finite machines, address tables and required bandwidth. However, NoCs are also required to be robust to traffic changes which can be achieved with correct routing scheme. For example, if traffic has lots of variation, dynamic routing scheme may be more suitable than static scheme, even though static scheme may be more suitable if traffic patterns are well known. [1 p. 169-172]

Routing scheme can be categorized as static when the path from source to its destination is predefined, or as dynamic when the decisions of routing are made at each switch that is based on the current state of the network. The down side of static routing is that it does not take in account the current state of the network which may cause an unbalanced load, although the amount of required logic is lower and the used algorithm

is simpler which is due to reason that packet reordering is not needed. All in all, if the transactions are well known and steady, the static routing is quite suitable. Furthermore, routing can also be divided into distributed and sourced. Distributed routing uses intermediate port tables or otherwise it would be required to calculate specific port addresses on the run. For example, to reduce the size of routing tables, XY-coordinates of the port can be given with the destination address. In this situation, routers would first compare the destination address with routing tables and if there is no correlation it would use a routing function to calculate the corresponding path. However, this may limit available network topologies and usually only destination addresses are preferred due to the higher logic area. In addition, source routing bases on its packet header which contains an extracted routing path to the desired destination. Packet headers are built inside NIs and do not use any information from routers. The positive side with source routing is that network tables and calculation functions are not needed but the down side is that packet header is larger and routing tables are needed on the source side. [1 p. 170-172]

2.2.4. Congestion- and flow control

Congestion is an event where the resources of the network are being used by multiple sources which leads to the delay of traffic which causes performance reduction in sense of delayed and rerouted packets. In addition, rerouting takes effectively more time which makes latency and bandwidth requirements harder to be achieved and to avoid such events, congestion and flow control are required. Figure 8 [1 p. 179] shows that flow control occurs between slave-master pairs, but it is also used between routers at the link-level. Figure 8 also shows that congestion occurs when transactions attempt to use the same router at the same time which in this case blocks the possible transactions between master 3 and slave 3. Furthermore, congestion control can be divided into closed and open loop networks which may or may not contain resource reservations. The down side of congestion avoidance without the resource reservation is its worse fit with hard timing requirements. [1 p. 179-183]

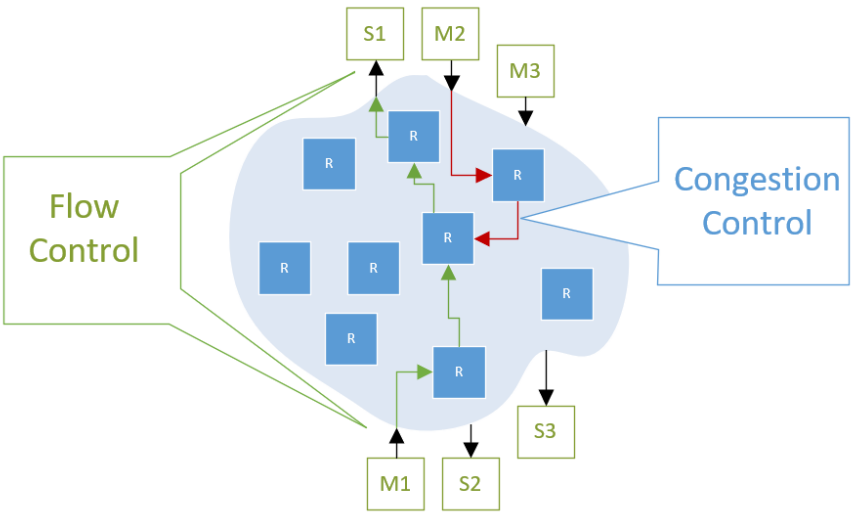

Figure 8. Locations and differences between the flow- and congestion controls.

A problematic issue with NoC architectures is that they may lead to the problems such as deadlock, livelock or starvation. These may lead to the event where the data is not correctly delivered, although it can be fixed with some sort of recovery system. Deadlock situation occurs when the whole data packet is being blocked by an intermediate resource, such as shared buffers, and to avoid deadlocks resource preservation could be used. For example, all required buffer arrays could be preserved from the start of transmission till the end and no other data packets could interfere with the transmission. Other solution could be deadlock avoidance where resources are allocated in advance of data packets. In addition, there could also be a deadlock recovery system where the resources of blocked transactions are released, and the blocked transactions are retransmitted afterwards. Starvation happens when a part of the data packet is blocked and lost, but the other parts have been passed correctly. A fix for the issue is to have a resource assignment scheme, like round-robin arbitration. In this case, buffers could be separated for the lower priority packets. Furthermore, Livelock occurs when data packet loops in a cyclic path and will not reach its destination which can be avoided either with the minimal data path lengths or with algorithms which reduce the probability of loops. All in all, there are many techniques to avoid these problems but those tend to increase logic area and cost, and many of them were developed considering macroscopic models which are not directly suitable for NoCs. [1 p. 39-41, p. 173]

The reactive methods for reducing congestion without resource reservation are packet dropping, dynamic routing, reducing the number of packets and informing other routers. With the packet dropping method all the transacted packets are deleted from the congested path and re-sent afterward. The downside of packet dropping is that it causes more traffic in a long run and it reduces effective utilization. Dynamic routing uses SAF and WH schemes in a way which only one packet at the time is sent to its specific output port and to maintain low latencies, wider inner-routing busses can be used. In addition, another way could be to inform other routers about congestions so that hotspots can be divided over the network. Packet limitation bases on the measurement of the maximum average latency which is used to tune the packet injection rate. [1 p. 179-183]

Congestion control methods which do include resource reservation are traffic scheduling and rate control. However, admission control and traffic policy must be in place to ensure successful traffic. Admission control books and reserves required resources, whereas traffic control keeps the injection rate below the given limits which were accepted by the admission control. Traffic scheduling is a method where the network does not guide traffic in a way which it is possible to collide. However, such scheduling requires knowledge of the propagation delays of the network and to have flow control. Traffic scheduling method has a lower cost than the others, but its overall latency is relatively high, although the worst-case latency would be as high as with rate control scheme which allows parallel transactions within boundaries. However, each router must be independent and non-blocking in terms of resource usage, and they do not interfere with the others. In addition, if transaction from different NIs share the data path they must not exceed its capacity and thus every NI must have its own traffic limiter. [1 p. 179-188]

Flow control can be divided into link-level and end-to-end flow control. Figure 8 shows the points of end-to-end flow control but there is also a link-level control between routers. All in all, the link-level is a control between single wires whereas the end-to-end control is between masters and slaves. [6 p. 4-5] Flow control is required

to limit traffic if the target network is not available or if it is blocked for some reason. Congestion control alone does not guarantee congestion free behavior, but it is required to avoid deadlocks. For example, if the target network is not available, the buffers of the transaction network are filled up, but the transaction would still be blocked and in the worst case, the traffic from other networks to their targets are also blocked. However, the idea of flow control is to avoid such behavior by denying master to transact for the specific slave. In addition, there are several non-resource reservation ways for flow control which are deleting old or new packets, returning packets and deflection routing. Deleting data packets is not commonly used in NoCs because its overall congestion is higher, although it could also work as congestion control. In returning method packets are forwarded back to its sender but the requirement would be that the sender accepts all returned data. With the deflection routing the packets are forwarded to another router where they are later transacted to its destination. [1 p. 188-189]

Flow control methods which require resource reservations are the types that ensure space availability at the slave network which can be done with end-to-end flow control. To check that there is enough space within the buffers for master to send, it can be done by using acknowledge / not acknowledge (ACK/NACK), STALL/GO or a credit-based response between the slave and master. These mechanisms can be seen in Figure 9 [1 p. 188-192]. The idea with ACKs is that the sent segments of the data packet are kept inside buffers which can be resend if the error (NACK) occurs. The NACK uses GO-BACK-N policy which means that the retransmission starts from the corrupted segment of the data and every segment after this is also resent. However, if the transaction went successfully, the stored segments from the buffer can be deleted. The STALL/GO flow controller requires two wires where the other goes forward to its destination and the other returns. Forward wire flags the data availability and the return wire flags whether the buffers are filled (STALL) or have free space (GO). The down side is that there is no error control. However, the idea of credit-based flow control is to send request from master to slave which slave may respond. Both require flow control of its own and the segments of data can leave master only if the request buffer of the slave has enough space. An example structure of credit-based flow control can be seen on Figure 9 c). The way a master and slave manage to control flow is to use a request and response credit counters which indicate the state of the buffer on the opposite side (2 and 4). Counters also keep care that the buffers 2 and 4 do not overflow. Sending a packet decreases its credit counter and receiving credits increases it. Credits are only being sent back when the opposite NI removes a packet from its response buffer. For example, when master sends a packet to the slave, it decreases master's credit counter by one and master may keep sending packets as long as its credit counter is above zero. When the slave removes a packet from its buffer (2), it sends a credit back to master which increases master's credit counter by one. The same mechanism works from slave to master as well. However, the credit-based control decreases bandwidth usage, for example by 30%, but it can be compensated by increasing slave buffer sizes which also increase the cost. [1 p. 132-135, p. 188-192, 6 p. 21-22]

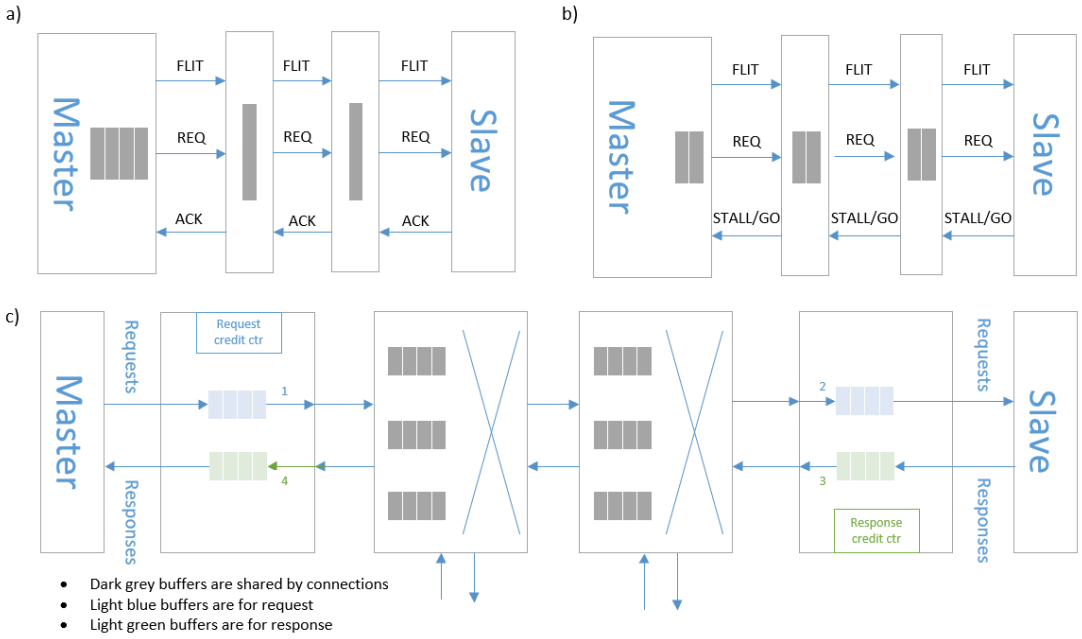

Figure 9. Flow control methods a) ACK/NACK, b) STALL/GO and c) credit-based.

2.3. Methods for designing NoC

The design of a NoC might require several configurations due to the difficulty of design choices and to aid designing, there are several CAD tools available. Issues which need to be solved before the completely working NoC are to analyze and characterize traffic, synthesize the topology, map and bind components within NoC, define traffic paths, resources and architectural parameters and verify the correct behavior of NoC. Most of the design steps can be solved parallel to achieve more efficient designing with feedback. Furthermore, CAD-tools can be divided into analysis and simulation, synthesis and optimization, and as toolkits for bus designing. [1 p. 323-325] More of those design tools and researches can be found in [1 p. 333-343]. In addition, there are also vendors, like Sonics [9], Arteris [10] and ARM [11] which combine the whole work flow into single simulation environment and Synopsys [12] which provides libraries to build such networks.

Designing is preferred as with layered structure that can be seen more clearly on Table 1 [1 p. 324]. The first simulation could be done at the higher-level where mostly bandwidth and delay-hop are considered, meaning that the topology synthesis, mapping, routing, resource reservation and architectural parameters should be decided at this level. However, the difficulty is to have accurate traffic patterns, power consumption and performance which corresponds well with the lower levels. The next lower layer is packet-level simulation where dynamic effects like buffer size, arbitration and routing policies can be observed. However, the traffic and components used should match with the actual hardware. After packet-level simulation, transaction level simulations can be made with HDL languages such as SystemC and architectural parameters can be tuned and minor changes to topology can be made. Lastly, cycle-accurate RTL simulations can be done to verify performance and validate the system. The results should verify performance and the system validation of the network. [1 p. 323-325]

Table 1. Table represents layered flow design for the NoC solutions

	Design phases	Models/effects	Key issues
High-level specification	Topology design, mapping, routing, narrow parameters	Analytical models, static effects, large solutions space	Accurate traffic, performance, power modeling
Packet-level simulation	Buffer sizing, arbitration and routing policy	Dynamic, fast C++ simulations, stochastic traffic	Accurate network & traffic generator models
Transaction simulation	Narrow parameteres, key topology changes	Dependencies in communication	Cycle accuracy, time consumption
Cycle-accurate simulation	Performance test, very few changes	Completely accurate	FPGA emulation, time consumption

2.3.1. SonicsStudio® Director

Sonics offers several non-blocking NoC solutions [9] which targets different objectives such as high performance and power efficiency. These solutions have support for Advanced High-Performance Bus (AHB) [13], Advanced Peripheral Bus (APB) [14], Advanced Extensible Interface (AXI) [15], AXI Coherency Extensions Lite (ACE-Lite) [15] and OCP protocols. Different NoC IPs are built under Graphical User Interface (GUI) based SoC development environment which is called SonicsStudio® Director. It eases designing with automation and traffic simulations which indicate possible bottlenecks and difference between design choices. For example, it is possible to combine several network types to work together and gain flexibility. There is support for scripts, SystemC and RTL level optimization, logic synthesis and analysis. The design tool can also generate Universal Verification Methodology (UVM) based testbench for functional verification. In addition, there are several error handling methods and integrated power and clock management which eases cross domain designing. [9]

SonicsStudio® Director includes SonicsGN® which offers router based, serialized and a high-speed network with the packetized transactions. These ensure efficient gate count, scalability and reduce wiring congestion. SonicsGN® networks scales from IoT up to servers. SonicsSX® offers low power and latency usage with high bandwidth networks which use switches, memory interleaving and different network services. It is well suited for video processing and as the interconnection of graphics subsystems. There is also SonicsLX® that is a limited version from SonicsSX® which offers balanced solution with high performance, low latency and low power consumption with optimized area. There is a possibility to use the full crossbar or mixed topology which is specified for mid-range IPs and SoCs. Sonics3220^{TM} is for non-blocking network solutions which are for long distances with the large number of IPs which are required to be power efficient with low latency. This network supports APB and OCP protocols and is meant to isolate slow speed I/O from high speed SoC. In addition, there is SonicsExpress^{TM} network IP bridge for the asynchronous clock, voltage and power crossings. Furthermore, the tool also has MemMax® memory scheduler for

Dynamic Random-Access Memories (DRAM) and SonicsMT^{TM} for performance analysis and debugging. [9]

2.3.2. Arteris FlexNoC

Arteris is an IP vendor which provides FlexNoC [10] design tool for customized NoCs. The design tool includes fundamental IP libraries for the NoC units, the exploration of various network topologies, compilation software to configure and generate synthesizable RTL blocks. There are also possibilities to run scripts and use GUI to design and simulate networks. The tool also uses engineering change order (ECO) to speed up the design flow which avoids wiring congestion in RTL phase, instead of in the post-layout phase. This reduces time usage in layout-phase and is generally more efficient. Traffic itself is divided into transaction, transfer and physical layers and the communication between different layers are handled inside Network Interface Units (NIU). This eases integration of different IPs to NoC regardless of used protocol since those are converted to internal streamlined protocol. FlexNoC supports protocols like AXI, ACE, AHB and APB and standards such as OCP, Processor Interface (PIF) and Basic Virtual Component Interface (BVCI). In addition, FlexNoC provides high utilization of wires. For example, the tool can generate more wires where required, for example near Central Processing Unit (CPU) and cache, and less wires with longer paths which also require lower bandwidth, such as Universal Serial Bus (USB). [1, p. 351-352, 10]

FlexNoC optimizes the insertion of repeater registers which eases to lower the power consumption. Congested paths can be cleared for the higher priority transactions that can be achieved with end-to-end QoS and rate regulators can be used to limit the bandwidth of certain sockets. In addition, tool supports traffic profiles such as latency sensitive, latency critical, real time, bandwidth sensitive and best effort. There is also a fine-grained pipeline insertion tool for making timing problems easier to detect and fix without affecting the other parts of SoC. Furthermore, FlexMem Multi-Array Memory scheduler can be used to connect any DRAM front-end controller to maximize utilization, reducing gate count and decreasing latency and wiring congestion. In addition, there is a built-in power and clock management which supports multiple domains to be used. For example, globally asynchronous and locally synchronous clock bridges are provided among with automatically inserted voltage level shifters. [10]

2.3.3. ARM CoreLink Network Interconnect

The Arm CoreLink Network Interconnect (NIC) [11] is a highly configurable design tool which offers high performance and optimized connectivity for AMBA protocols. It has NoC-like behavior which provides connectivity from single bridge up to 128 masters and 64 slaves which are combinations of different AMBA protocols. Data is forwarded through interfaces with switches that can be in 32 bits up to 256 bits wide. Forwarded data can be packetized within the buffers which allows transactions between the different bus widths. This allows designers to minimize the number of wires, used area, and to have desired performance. All in all, the tool consists of several libraries which are Network Interconnect (NIC), Advanced QoS, QoS using Virtual Networks (QVN), Thin Links (TLX), AMBA Domain Bridge (ADB), AXI-to-AHB

Bridge (XHB) and Low-power distributor (LPD). There is also a CodeLink Creator which uses algorithms to accelerate interconnection designing which is then combined in Socrates DE to provide integrated environment. The environment speeds up the designing of the whole interconnection. The tool also includes full functionality AMBA 4 AXI4, AXI3, AHB-Lite and APB interfaces. [11]

Network Interconnect provides flexibility in form of register placement that can be used where needed which allows fine-grain tuning between latency and clock frequency. Advanced QoS provides efficient and intelligent traffic management by using regulators which are controlled with dynamic bandwidth and latency. Virtual Networks prevents cross-streaming or head-of-line blocking by using the priority allocated buffers for different virtual channels which is used in both the interconnect and dynamic memory controller. Thin Links library is used to reduce wiring congestion and ease with timing closures. For example, AXI4 can be packetized and its data can be transmitted through switches with fewer signals. There is also possibility to use Cache Coherent Network (CCN) or Interconnect (CCI) to be extended IO coherency to masters. [11]

2.3.4. Synopsys DesignWare IP

Synopsys DesignWare IP [12] is an automated solution which provides GUI to design interconnections that are configurable and flexible and have reduced complexity and improved productivity in used time-basis. Its interconnection structure is closer to Multi-layer bus interconnection than NoC solutions and it only supports protocols such as ARM AMBA 2.0, AMBA 3 AXI, AMBA 4 AXI and ACE-Lite. Furthermore, DesignWare IP contains Discovery Verification IP (VIP) for AMBA protocols which provide methodology, verification and productivity features that ease to achieve verification convergence. VIPs are integrated with Protocol Analyzer which also provides UVM sequence library, a testbench support for Verification Methodology Manual (VMM), UVM, Open Verification Methodology (OVM) and Verilog. It also has performance checking, a configurable interconnect model, debug port for the transaction tracking on waveforms, reference verification platform and extensive callbacks and messages. More of Synopsys DesignWare IP solution can be found in [12]. However, they are briefly discussed below. [12]

DesignWare IP offers high-performance and low-latency interconnections for AMBA 2, AMBA 3 APB and AXI and AMBA 4 AXI protocols. AXI provides a hybrid architecture with reduced area and power consumption and with lowered routing congestion. There are also advanced high-frequency pipeline options, arbitration, interleaved transfers, full ID ordering mode, bi-directional commands and memory maps. Furthermore, there is an interconnection support up to 8 AHB layers to connect one AHB slave with static layer arbitration, external priorities and starvation prevention is available in a manner of RETRY. In addition, DesignWare IP has support for high-performance and low-latency interconnection bridges between AMBA 2 APB and AHB protocols with narrow bus conversions and interrupt vector controllers. There is also a bridge support for AMBA protocols which can be tuned to be performance efficient with low-bandwidth or high-performance with high-bandwidth. These bridges have support for cross-clocking, endianness, ordering, arbitration schemes, bus size matching, configurable buffer depths, pipelining, optimized synchronous-, asynchronous- and optimized single clock operations, interleaved transactions and full ID ordering mode. In addition, there is support for configurable

store-forward and cut-through modes and custom interconnections to AXI protocol with 100 percent throughput efficiency. [12]

DesignWare IP also offers AHB Direct Memory Access (DMA) controllers which are highly optimized. There is also a flexible, multi-interfaced and centralized AXI DMA controller which is AMBA 3 and 4 AXI compliant and can have AHB, AXI4-Lite or APB3 slaves. There are also generic APB peripherals for I/O control which supports both hardware and software configurations. For advanced use, protocols such as Serial Peripheral Interface (SPI), Synchronous Serial Port (SSP), Inter-Integrated Circuit (I2C), Universal Asynchronous Receiver-Transmitter (UART) and programmable microwires are supported with APB peripherals. Also, there is time configurable and an interrupt supported watchdog timer. [12]

2.4. Power and performance

It is important to balance between power, area, VLSI resources, performance and robustness to environment changes. For example, power consumption can be affected by varying between different transaction paths. NoC is basically built from finite state machines, address tables and wires which increase when the complexity grows higher. It is also important to notice that the performance might suffer if the routing scheme is not suitable for the used hardware. On the other hand, traffic scheme may work very well but if the changing to other is done poorly, it causes a decrease of robustness and overall performance suffers. [1 p. 170-171] The cost of using NoC comes from increased silicon area and number of active logic. For example, static and dynamic power consumption increases along with the total cell area but whether the power consumption is over given limits, it is up to chosen topology. On the other hand, if the goal is to have low power consumption it may mean higher latencies. [1 p. 33]

Latency can be defined as how long it takes for the transaction to finish which is measured in clock cycles. Where the latency indicates time, the bandwidth indicates physically limited speed, in bits per seconds (bit/s), for the specific channel. However, when the latency and bandwidth are known, the throughput of the specific channel can be calculated by multiplying the number of transactions with bus width and frequency and dividing it with the whole latency of transactions. Theoretical throughput can be calculated with Equation 1 where the total latency is in clock cycles, f_{clk} in Hz and bus width in bits.

$$Throughput = \frac{Number\ of\ transfers}{Total\ latency} * f_{clk} * bus\ width \quad (1)$$

If the total latency was decreased or the bus bit width and frequency were increased, it would make the throughput higher. For example, a single AHB-Lite transaction takes the minimum of two clock cycles, but such protocol supports pipelining which means that transactions from a master can overlap and thus decrease the overall latency. This would increase the throughput and would be the more efficient way to transfer data. In addition, the performance increase would also happen if bus widths were multiplied by two, but the power consumption would be higher. However, the transfer latency of protocols is not the only latency that needs to be considered since there is also an additional latency from arbitration, synchronization, bridging, masters and slaves. [16]

With Complementary Metal Oxide Semiconductor (CMOS) logic there are dynamic and static power consumption to be considered. Dynamic power consumption

consists of switching activity that can be divided into the switching- and short-circuit and its consumption is related to the activity of design, regardless of whether it is intended or glitched. However, the dynamic power consumption can be reduced by the careful system design, for example by changing factors such as load capacitance, voltage levels, switching activity and clock frequency. Static power consumption is formed from leakage current through transistors in its static cut-off state which can be affected by changing the substrate doping and the physical level of design topology. For example, when the physical dimensions of transistors are scaled down it usually requires lower supply voltage levels to keep dynamic power consumption within the reasonable limits. However, lowering voltage levels also increases static power consumption but it can be reduced by using techniques such as transistor stacking, multi-threshold CMOS or dynamic threshold Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFET). [17]

Equation 2 shows how power consumption can roughly be calculated and Equation 3 shows roughly what parameters affect the most. In both equations, dynamic power consumption is inside the parentheses.

$$P_{tot} = (P_{sw} + P_{sc}) + P_{stat} \tag{2}$$

$$P_{tot} = \left(C_L V_{dd}^2 a f_{clk} + \frac{\beta}{12}(V_{dd} - 2V_T)^3 \tau a f_{clk} \right) + I_{leakage} V_{dd} \tag{3}$$

Equation 3 shows that load capacitance C_L, supply voltage V_{dd}, clock frequency f_{clk} and design activity factor a have high effect on dynamic power consumption and for this reason these are commonly modified. However, the gain factor β, threshold voltage V_T and symmetric rise and fall time τ can also be changed. In addition, the static power consumption can be affected with supply voltage changes, but its leakage current is more like a side effect of the used standard library. [17] More precise explanations of parameters can be found in [17].

2.5. Estimation based on researches

One of the early implementations of NoC design tools was NetChip which was based on Xpipes library, XpipesCompiler, SUNFLOOR and SUNMAP. XpipeCompiler was used to generate parametrized SystemC IPs and SUNMAP was used to select topology, map IPs and to gain an early power and area estimation. SUNFLOOR was used to synthesize, optimize performance and to gain an estimation of the area and power consumption. The design tool also supported many features which were explained in this thesis earlier. For example, support for parametrized switches, data widths, buffer sizes, pipelines, flow control, routing, error handling and standard interfaces such as OCP. [1 p. 332-337, 18, 19] Table 2 shows example results from research [18] which were achieved with the tool. At the examples there were a few high-end video applications such as picture-in-picture (PIP) and multi-window display (MWD) with maximum link bandwidth of 500 MBps. PIP had 8 cores whereas MWD had 14 cores and both used process technology of 100 nm. [18]

Table 2. Video applications PIP and MWD were built with the NetChip tool

	Picture-In-Picture (PIP)			Multi-Window display (MWD)		
Topology	**Hop**	**Area (mm^2)**	**Power (mW)**	**Hop**	**Area (mm^2)**	**Power (mW)**
Mesh	2.1	21.06	201.47	2.1	70.32	321.99
Torus	2	30.72	204.62	2	76.80	323.56
Hypercube	2	30.72	208.02	2.35	80.24	341.62
Clos	3	29.18	208.16	3	78.96	333.29
Butterfly	2	24.46	204.76	2	68.64	329.70

The result differences of Table 2 were explained with the different number of routers, their link lengths and different router sizes. For example, mesh topology was more symmetrical than the butterfly which allows better layout and used partly smaller switches. Furthermore, there was also a video object plane decoder (VOPD) simulated with mesh topology. It was observed that with the customized network the design tool was able to optimize the required number of switches which resulted in 5.73 area reduction ratio and 2.71 power reduction ratio. [18]

Importance of minimizing the overall wire length of interconnection can be explained via its parasitic resistance R_T, capacitance C_T and inductance L_T. For example, when wires are physically narrower, longer and are grouped closer to each other, they do have larger parasitic components which increase power consumption and propagation delay. Such issues cause wirings to bottleneck the performance and repeaters and pipelines are required for achieving the timing constraints. [20 p. 45-56] Table 3 contains throughputs and leakage power numbers which were determined with Advanced Design System (ADS) tools at the research of [20 p. 101-119]. Upper throughput numbers determine achieved values without the process parameter variations and grayed numbers below indicate the highest achieved throughputs under the process parameter variation. However, with the mean leakage power, the idea is same but now the grayed areas indicate the standard deviation. The symmetrical changes of throughputs were explained with the clocking network and that each bus of the router was 8-bit wide. Furthermore, when smaller process technologies were used, the routing delays did increase with larger deviation. In addition, the wire delays were a few hundred picoseconds up to nanosecond and the delay order of topologies were from the slowest as octagon, torus, folded torus, BFT and cliché. [20 p. 101-119].

Table 3. Different synchronous topologies with process technologies of 65, 45 and 32 nanometers. Grayed lines indicate the effect of the process variations

Topology	Throughput (Gbps)			Mean leakage power (mW)		
	65 nm	45 nm	32 nm	65 nm	45 nm	32 nm
Octagon	86.8	87.42	89.21	9.47	11.82	38.83
	79.9	*77.69*	*66.68*	*1.79*	*2.89*	*10.95*
Cliché	86.75	87.41	89.21	5.85	6.53	12.65
	79.86	*77.69*	*66.68*	*0.60*	*0.90*	*2.64*
Torus	108.4	109.3	111.5	10.26	12.30	37.57
	99.83	*97.11*	*83.34*	*1.45*	*2.29*	*8.03*
Folded torus	108.4	109.3	111.5	9.39	11.45	23.26
	99.83	*97.11*	*83.34*	*1.08*	*1.81*	*3.97*
BFT	43.38	43.71	44.61	3.42	4.19	9.54
	39.93	*38.84*	*33.34*	*0.61*	*0.81*	*2.94*

Common to all topologies, but butterfly fat tree (BFT), was that each router had its own NI attached. However, the butterfly had NIs only attached at its lowest roots. This means that the butterfly topology had 16 NIs and 6 routers whereas the other topologies had 16 NIs and 16 routers. Furthermore, the octagon topology had two networks attached to each other via one link and the cliché network was like 2D-mesh network from Figure 3b) and torus was like from Figure 4a). However, the difference between folded torus and torus was that routers were divided partly to the folded side which causes link lengths to be more symmetric and longer. [20 p. 75-79]

Another research [21] used network simulator 2 (NS2) tool for estimating different NoC architectures which were earlier described in Figures 3-5 and are shown at the Table 4. The table contains rough estimations about throughput, latency and packet drop probability. The research found out that the SPIN and octagon topologies had the highest throughputs which were approximately 2-3 times faster when compared with the cliché, folded torus and BFT topologies. Research also found out that the latencies behaved similarly with throughputs, although latencies were observed to have more dispersion. The BFT and folded torus topologies seemed to have approximately two times more latency than the octagon and SPIN, but the cliché had three times more. The lowest latencies with the SPIN and octagon were explained by the number of required hops between switches. Furthermore, there was also packet drop probability observed which indicated that BFT had the lowest probability due to existing alternative data paths. Simpler structures like the octagon and SPIN suffered due to the lack of different data paths and the packet drop probability increased radically at the beginning. [21]

Table 4. An overall comparison between different topologies

Topology	Throughput	Latency	Drop probability
SPIN	Highest	Low	Highest
Octagon	High	Lowest	High
Cliché	Medium high	Highest	Medium high
Folded torus	Medium low	High	Low
BFT	Lowest	Medium high	Lowest

3. IMPLEMENTED VERSIONS OF NETWORKS

The purpose of several implemented networks was to achieve comparable results with the reference multi-layer bus interconnection and to gain an overview of how a NoC design tool operated. For example, what would be the main differences with created architectures and how these optimized NoCs did handle clock bridging, protocol conversions and the traffic itself. In addition, it was interesting to observe how generated networks did scale up.

3.1. Specifications of networks

The main goal was to achieve reliable results with high throughput, low latency and small area and power consumption. However, parameters such as frequencies, bus widths and protocols were pre-defined which limited maximum throughput and minimum latency. For example, AHB3, APB3 and AXI4 protocols were used but NoCs did not support all the features and some additional options were enabled, such as support for AXI fixed burst translation into AHB single transactions. Figure 10 shows color-coded connections between inputs and outputs, clock frequency ratios, protocols used and bus width ratios. For example, the bus widths of masters 1-5 are four times wider than master 6 and clock frequencies for masters 1-5 are synchronously half from the maximum clock frequency that master 6 uses. Masters 1-5 are identical to each and all can be thought as five separated inputs. The same applies with output slaves 2-4 and 5-8.

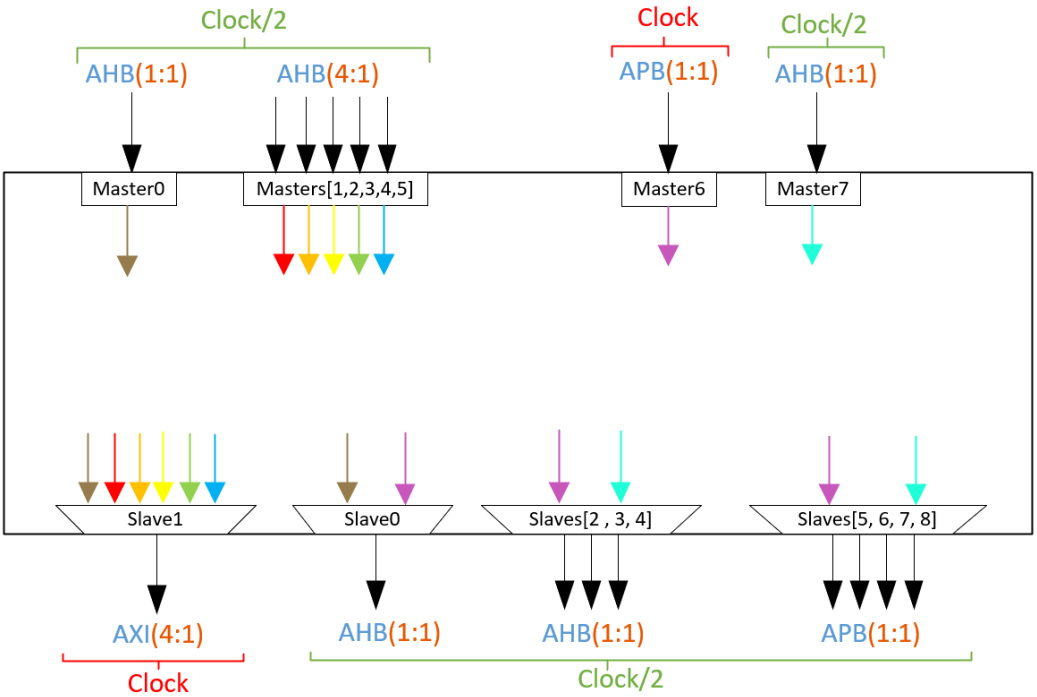

Figure 10. Common specification for the comparable networks. Ratio within the parentheses indicates the proportional bus widths.

Figure 11 shows more complex network connections which were tested and where masters 8, 9, 10 and 11 were included along with the slave 9. The reason for this network was to research how well the NoC did scale up when more traffic was introduced to network. However, for increased complexity the NoC was not fully comparable with the reference network but its performance versus the simpler version were observed.

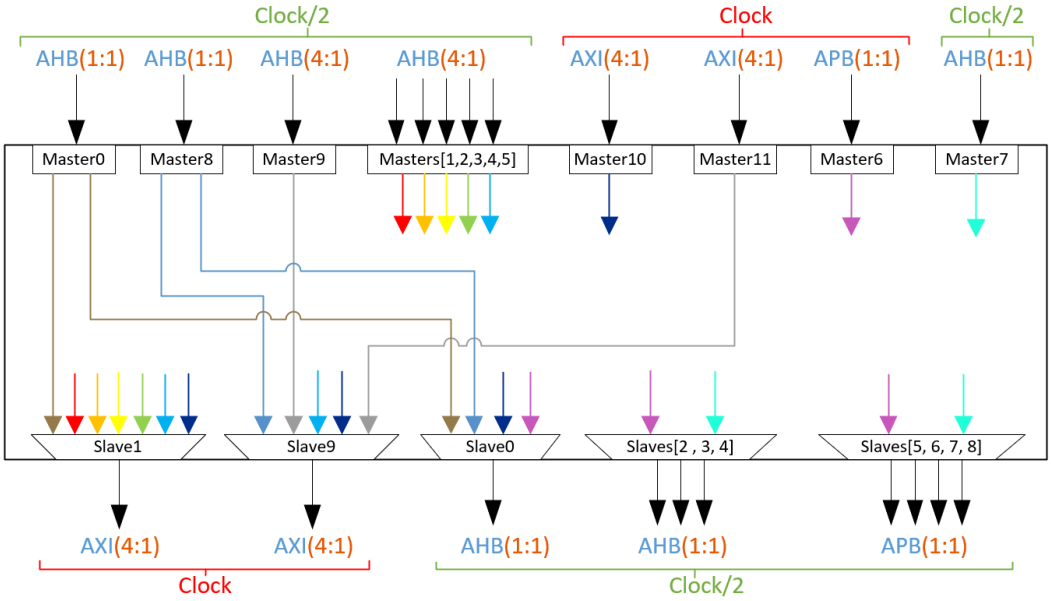

Figure 11. Common specification for the complex networks. Ratio within the parentheses indicates the proportional bus widths.

Common specifications for the networks introduced in Figures 10 and 11 were that AXI masters and slaves were capable of bursts whereas AHB masters and slaves were not and their transactions had to always use handshake signals or otherwise an error would occur. Furthermore, networks supported segmented address memory mapping and every segment could be read or write accessed. The power domain was shared over the whole network and its clock domains were 2:1 synchronous to each other with a common reset. Bus widths were with the ratio of 4:1 as it can be seen in Figures 10 and 11. The main differences, on the other hand, were that the design tool generated NoCs supported rotated arbitration but with the reference network it was based on input port indexing. In addition, generated NoCs used AHB3 protocol with selected options whereas the reference network was based on AHB-Lite protocol.

3.2. Reference Multi-layer bus interconnection

The reference network was created by connecting multiplexers, protocol conversion IPs, bus width conversion IPs and clock bridges together. The high-level model can be seen in Figure 12. The structure was relatively simple where most of the logic was at the lower clock speed since the clock bridges were located at the very end of data paths. Bus widths, on the other hand, were 4:1 for Mux 2 and 1:1 for Mux 3 and mainly AHB-Lite protocol was used for data delivery in both multiplexers which was converted to the desired protocols at the end of data paths.

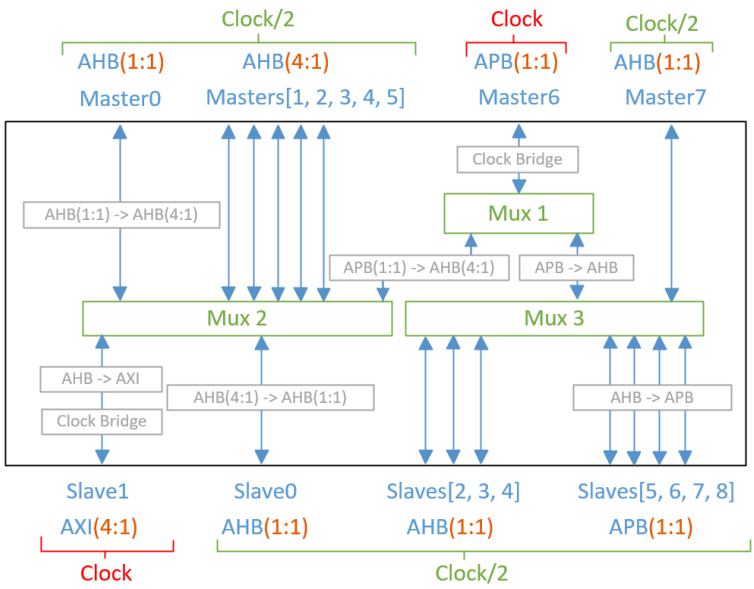

Figure 12. The reference multi-layer network. Ratio within the parentheses indicates the proportional bus widths.

The reference network supported index-based arbitration which caused an issue when high performance was desired. For example, the higher priority inputs were capable to choking lower priority transactions. Memory maps were in multiplexers which then granted accesses to different inputs. However, it is important to notice in Figure 12 that there was no logic at inputs and outputs but rather simple wire interfaces for IPs outside the network.

3.3. Created network on chip solutions

The design tool generated networks were built by first defining all the required parameters and then generating connections by initiating switches where desired. Latency and throughput were also affected by the defined data bus widths for internal protocols. In addition, a quick higher-level testing was made to gain an overview of performance. For example, where the bottlenecks were located and what were the throughput and latency of certain input-output pairs. However, to generate optimized networks, it was required to work through a few iterative loops to gain the best results, although the first tryout did provide relatively good results. The performance could have been increased for specific master-slave pairs by enabled QoS, but it would have benefit mostly with the parallel transaction and would have been tradeoff between performance and increased logic area. All in all, it was generally more efficient to modify switches, buffers, bus widths, arbitration and clock frequencies before the QoS.

Commonly created NoCs were considered as a soft core with a fat tree topology which were an irregular, application specific and pre-defined. Networks were also considered as an indirect which used source routed packet switching, although there was only one data path to its destination. In addition, deadlocks, starvation and livelocks were prevented due to not existing cyclic paths, resource reservation and arbitration. There were also handshake signals and error responses for indicating the unsuccessful transactions. However, no data limiters were used which led to situation where bandwidth was equally divided among all inputs of a switch. Table 5 represents

summary of created NoCs that were tested. Only the versions one and two were comparable with the reference network since the other versions had additional inputs. However, NoC versions 3-5 were compared with NoC versions 1 and 2, to observe the difference with performance and logic area.

Table 5. Summary of the created NoC versions

	NoC version	Latency	Header sent by
1	Simpler & faster	Low	Separated wires
2	Simpler & slower	High	Data bus wires
3	Complex & faster	Low	Separated wires
4	Complex & slower	High	Data bus wires
5	Complex & slower & QoS	Low	Data bus wires

Figure 13 shows the simpler version of generated NoC structure which was compared with the reference network. Due to its simple structure with clock frequencies and connection map, there were only two switches required. However, two versions of simpler NoC were created and the purpose of these versions was to gain an overview of performance and resource tradeoff. Thus, the other version was for the minimum latency with larger wire count and the other for higher latency with minimum wire count. The 1st NoC in Table 5 was the version of minimum latency where switch 1 had the bus width ratio of 4:1 and packet header were sent by using parallel wires. For the higher latency version, all switches were with the ratio of 1:1 and packet headers were sent by using the common busses. However, bus width splits from 4:1 to four serial transactions did increase latency which had already been increased by a few clock cycles because of shared busses for the packet header delivery. The higher latency NoC was the 2nd in Table 5. However, it is important to notice in Figure 13 that protocol conversions to the internal protocol did occur in every master and slave, and that arbitration was enabled as rotate for every switch.

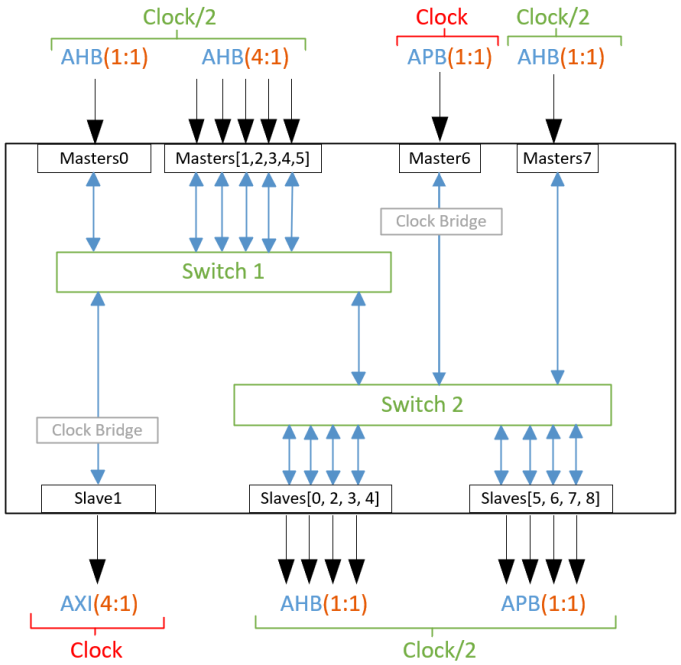

Figure 13. A simpler design of the tool generated NoC. Ratio within the parentheses indicates the proportional bus widths.

Other three different NoCs were created based on Figure 14. In the 3rd version, the minimum latency was achieved with wider busses and with separated packet header wires. In this network, the clock frequency was set as the highest and bus widths were designed with the ratio of 4:1 for the switches 3 and 4 which were also colored as red in Figure 14. The 4th version of NoC was created like the previous simpler higher latency version. Thus, the bus width ratios were designed as 1:1 and packet headers were sent via common busses which led to the increased latency and to the minimum wire count. However, clock frequencies were similar as with the previous complex version. Furthermore, a version of NoC with lower latency and QoS was created but the QoS was only included for AXI protocols.

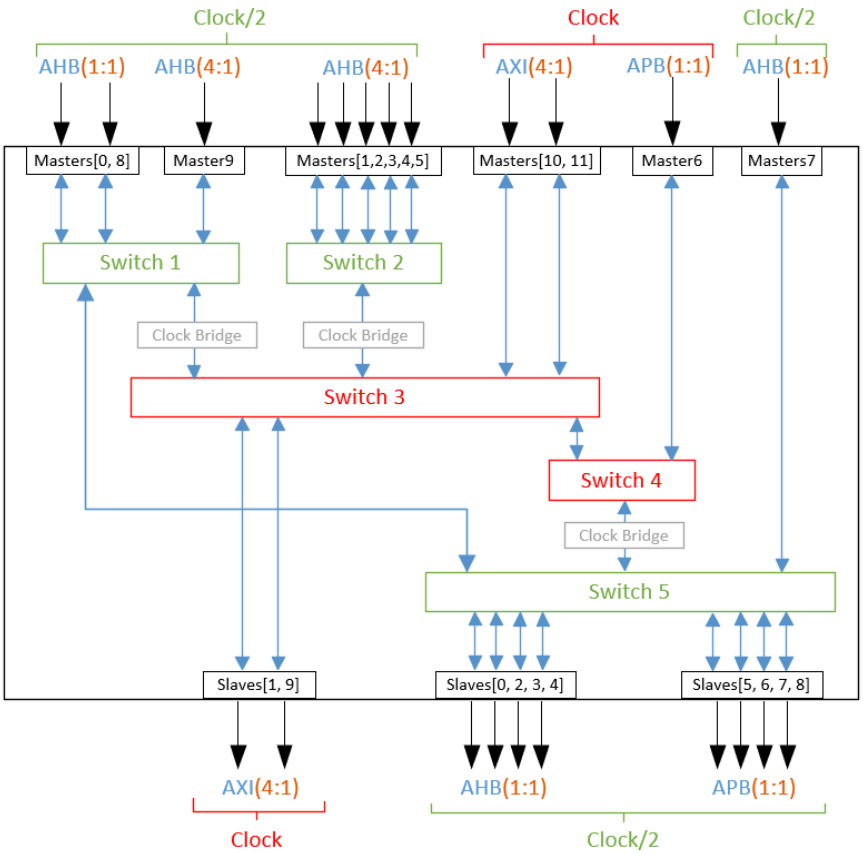

Figure 14. A complex version of the tool generated NoC. Ratio within the parentheses indicates the proportional bus widths.

3.4. Main differences of designing paths

Designing NoC with a tool was relatively fast when compared the situation where all testing and implementations were handcrafted. However, tuning all the parameters and architectures, filling higher level testcases, optimizing performance and verifying basic functionality at the RTL level, it took approximately a few months to create all versions of the tested NoCs. The time consumed also included the learning curve of the design tool itself, although it was still faster than completely to build a network from the scratch. However, on the top of creating the RTL code, verification at the lower level must also be done but since the design tool already provided tested and

working RTL codes, it was not really needed to create tests for the precise verification at the lower level but rather that all the IPs communicate correctly. With the self-crafted networks, it was more important and time consuming to test the whole network and verify its functionality in every test case. In addition, with the handcrafted situation there were still these iteration loops for the optimized solution.

When the complexity of networks increased, it became more difficult to achieve all performance requirements, although those were easier to be achieved with the tool since the higher-level testing exist. In addition, architectural modifications were rather quickly made with the tool when the changes such as additional NIs, connectivity remapping or the performance enhancements were required. Furthermore, the RTL code was quickly regenerated after changing some parameters and complete verification was not needed. However, with the handcrafted situation, the changes might also be rather quickly made but the verification needs to be done all over again. With the design tool, it took approximately 15 minutes up to a few days to modify and test performance at the higher level, and usually there were no changes needed at the top-level RTL module and the simulations were quickly rerun.

The cost of using the NoC generating design tool was that the created networks had their backbone which cannot be much changed. Whether the desired networks were simple or complex, the tool handled it in the same way. For example, protocol conversions to the internal and backward were always made which may not be so ideal in every situation. This led to the architecture where it was more beneficial to design one large network, in terms of lower logic area and higher performance, instead of several smaller ones. These design choices were needed to take on account at the system level.

4. TESTING ENVIRONMENT AND RESULTS

The performance was measured by a testbench that contained different test patterns for each master which were controlled by VIPs. Traffic from the VIPs was monitored and information such as throughput, latency and number of transactions were gathered. The RTL modules of networks were placed in the real system where the synthesis was also run. All in all, performance and physical implementations were affected by the logic outside of the networks.

4.1. Test cases, environment and tools

There were 6 different test cases that were used to gain an overview of performance. For example, how well the networks did operate with different input latencies, what would happen if there were lots of parallel traffic or if there were only one VIP transacting at the time. Information from VIPs was collected by the monitors which measured time between the start and end points of each transaction and calculated the throughput based on the bus widths and measured latency. However, logic outside the networks caused its own effect to the achieved performance. For example, with the certain paths there was logic which caused additional clock cycles to the measured latency and thus its throughput, although it was considered that the external logic did not cause an uncontrolled bottlenecking to its traffic.

Figure 15 shows the used master-slave pairs and combinations of single write and read that were mainly used. Orange glowed masters 8-11 and slave 9 were for the complex NoCs which also had more traffic due to the added AXI and AHB interfaces. In addition, AXI master 10 writes and reads the bursts size of four beats to AXI slave 9. Furthermore, arrows directed to down indicates writes, up reads and both ended arrows were read and write combined. Masters 1-5, slaves 2-3 and slaves 5-8 were almost identical to each other inside its own block, where only the memory and connection mapping differed.

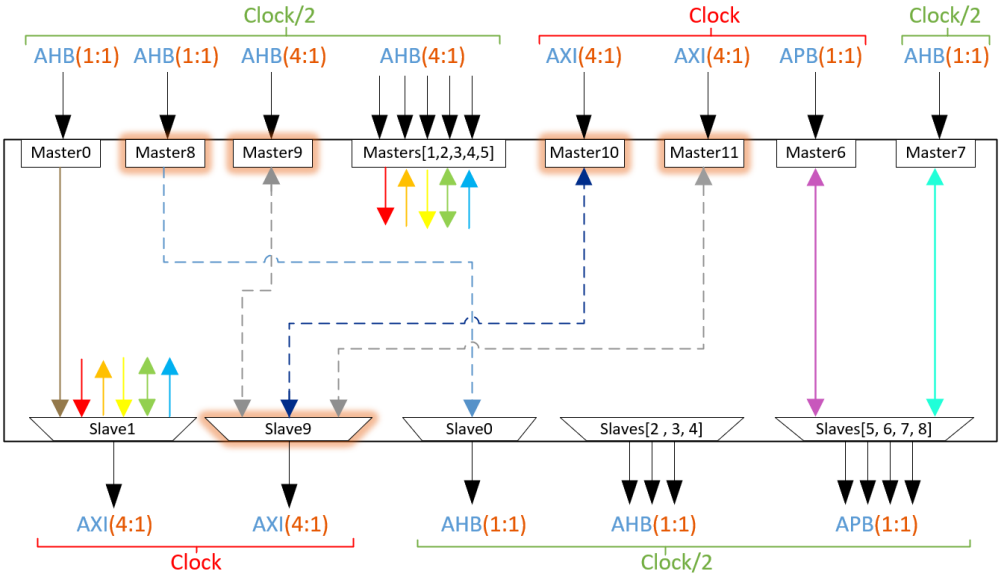

Figure 15. Visualization of the traffic and master-slave pairs. Dashed arrows and orange glowed masters 8-11 and slave 9 were added for complex networks. Ratio within the parentheses indicates the proportional bus widths.

Table 6 summarizes used test cases and Figure 16 shows a general view of test patterns which were used at each VIP. For example, the first test in the Table 6 can be thought as the zero delay in Figure 16. This meant that an additional clock cycles were not introduced between the data packets. However, in tests 2-5 the delay was set as 5, 10, 25 or 50 clock cycles which limited the theoretical maximum achievable throughput that were worsened because of the logic from the network itself and from test environment. In addition, the last test was to see the minimum achievable latency and highest throughput when a VIP sent all its transactions without being affected by the other transactions. It is also important to notice that tests 1-5 were stopped after 280 us since some VIPs had completed all their transactions before the others. For example, when 5 different masters shared the common bandwidth and when two of them finished their transactions, the rest of the masters did gain larger portions from the shared bandwidth. Such changes at the end of the test would have had an undesired effect on results, like the faulty average throughput of single masters.

Table 6. Used test cases to gain an overview of performance

Test	Description of the test
1	All VIPs are transacting with full speed
2	All VIPs are transacting with added delay of 5 clock cycles
3	All VIPs are transacting with added delay of 10 clock cycles
4	All VIPs are transacting with added delay of 25 clock cycles
5	All VIPs are transacting with added delay of 50 clock cycles
6	VIPs are transacting one by one with full speed

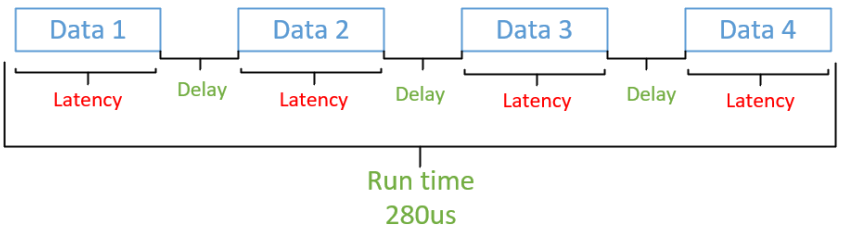

Figure 16. General view of VIPs test pattern and its run time.

The aim of simulations and synthesis were to achieve the overall guideline of performance, power consumption and required logic area. Thus, the most accurate synthesis tool was not required but rather that the achieved results did not have much analysis related inaccuracy. More information related to different RTL-level power estimation methods can be found in [22]. For the given reasons and resources available, RTL-level simulations were run with QuestaSim [23] and the gate-level synthesis were run with Synopsys Design Compiler (DC) [24]. Figure 17 shows simulation environment and synthesis [24 p.46] flows which were used in this thesis. For example, it shows how monitors and VIPs were connected to each other and how additional environment logic affected the performance. In addition, the figure shows that RTL design files, design and optimization constraints, operating conditions, Unified Power Format (UPF) and activity files were first read, and their correctness were tested. After checking design and timing issues the design went through HDL compiler and read Synopsys Design Constraints (SDC), technology and symbol libraries. Furthermore, design exchange format (DEF) files were also read to have a rough placement of the overall design. The next step was to work through the

optimization, timing closure testing and test synthesis by using a design-for-test (DFT) compiler [24 p. 42-49]. The result was an optimized netlist that was used to generate the gate-level design reports.

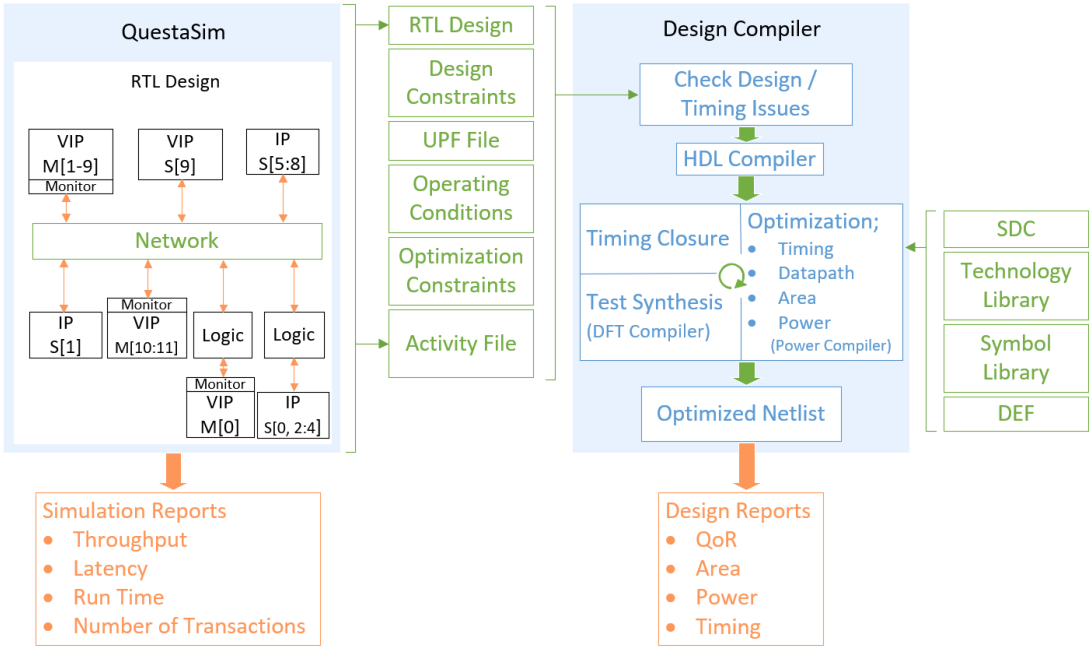

Figure 17. The simulation environment and synthesis flow.

The optimization effort for flattening were set as high for NoCs and activity files were generated from full speed tests. However, activity files were not used with the optimization of designs but to gain reliable estimations of the dynamic power consumption in its worst case. Static power consumption was also gathered along with the required sequential and non-sequential logic. Furthermore, technology libraries available were identical with each network. For example, low threshold voltage (LVT), high threshold voltage (HVT) and standard threshold voltage (SVT) libraries were used and the temperature corner point was kept unchanged. However, the usage ratio of libraries was left for the synthesis tool for being decided.

4.2. Performance of networks

The main performance differences over the networks were that the NoC handled parallel traffic more linearly. This was partly due to reason that the reference network did use index-based arbitration which blocked least prioritized ports in the worst case. For example, with the full speed test the average throughput over whole networks was a few percent higher with the reference network but with the additional delay of 5 clock cycles it dropped below the low latency version of NoC. However, when the traffic injected to networks decreased down to the additional delay of 25 clock cycles or higher, the performance was quite equal. Figure 18 shows the average throughput curves of tested networks, but the throughput ratios are different in each diagram. However, roughly the throughput of the multi-layer reference network was between the lower and higher latency version of NoCs.

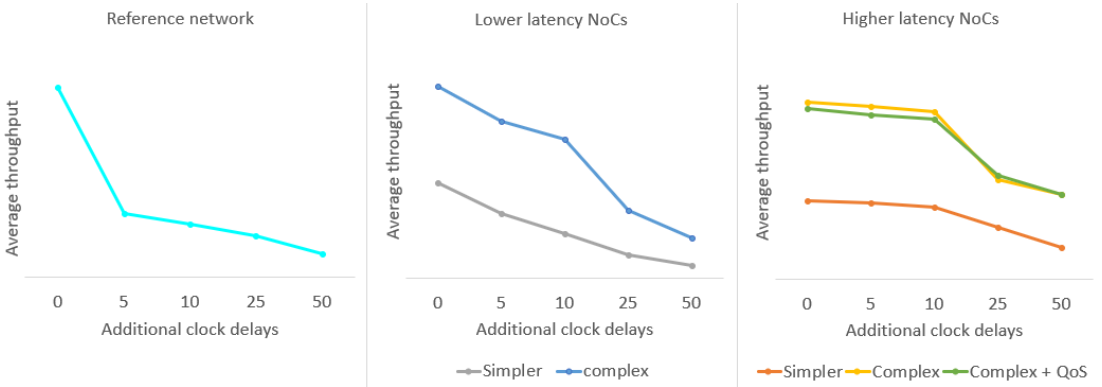

Figure 18. The average throughputs curves of the tested networks. Throughput ratios are different in each diagram.

The throughput drop of the reference network at the additional delay of five clock cycles can be explained by index-based arbitration. It was noticed that the lower priority inputs of masters 1-5 began to choke higher priority input ports. Since the traffic of low priority ports were also slower than the higher priority ports, the higher throughput transactions were choked which led to situation where overall throughput decreased. However, already slowed traffic had not been caused by the network itself but rather from external logic which means that each network suffered from the same response latencies. The affect from slower transactions could have been balanced by changing architecture but since the network was designed for specific purpose, it was not necessary to make changes. All in all, the architecture of the reference network suffers more from slow read transactions. For example, if all the transactions were fast writes, the average throughput would be much higher.

Figure 18 shows also that the higher latency NoCs were bottlenecking the traffic when the additional latency was below 10 clock cycles and that the shapes of throughput figures were similar. Such behavior was mainly due to the reason that throughput was limited by the architecture itself, although the throughput was about twice as high with the complex version. However, bottlenecking was not a huge problem with lower latency NoCs and the throughput grew somewhat linearly, even though the fastest paths like AXI busses were bottlenecking with the additional delay of ten clock cycles or lower. The addition of AXI busses also explains why the throughput increase was higher with complex NoCs between the additional delays of 10 and 25 clock cycles. However, above the additional delay of 25 clock cycles, the single transaction AXI bus began to slow the one with burst transactions. In addition, without AXI interfaces the throughput curves of complex NoCs would have been closer to simpler versions but with higher overall throughput. In Figure 19 the overall shapes of Figure 18 can be better understood since the throughput of the reference network and complex NoCs are split into group, although throughput ratios are different between Figures 18 and 19. Figure 19 also shows that when more traffic was introduced to NoCs it did not choke the other inputs, even though NoCs with the additional latency introduced by the architecture had some issues, but those were designed to behave in the certain way and worked as expected. The simpler versions of NoCs are not shown in Figure 19, although the throughput curves were like complex NoCs but without the green curve of AXI.

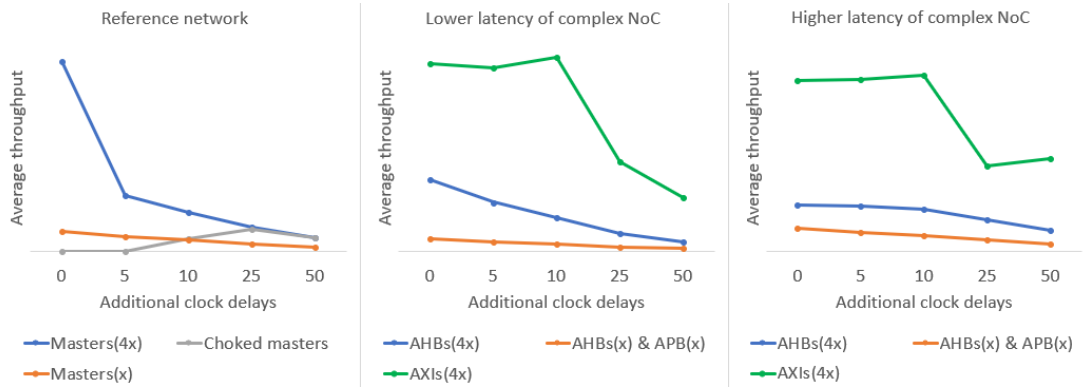

Figure 19. Average throughputs are split into groups of reference and complex networks. Throughput ratios are different in each diagram.

Differences were also seen when only one master was fed by traffic which corresponded the 6th testcase from Table 6. From this testcase it was monitored that the inputs between the reference network and lower latency version of the simpler NoC had throughput difference approximately from 0.8 up to 4.7 times in favor of the reference network. When the differences were summarized, the overall average throughput was approximately two times higher with the reference network. Furthermore, when the reference network was compared with the higher latency version of the simpler NoC, the throughput differences were 2-10 times higher. In addition, to support these findings latencies were also monitored. It was discovered that the average latency was 1.3 times higher with the simpler version of the lower latency NoC, which was 2.5 times higher with the higher latency version of NoC.

Performance differences could have been much higher if the transaction types were changed to support only fast writes since the reference network suffered slightly more from slow read transactions. The reason why the reference network did work better with fast transactions was due to reason that it did not use an internal protocol with its traffic. For example, it took a few clock cycles from NoCs to convert protocols forth and backward to the internal protocol whereas with the reference network was only the AHB to AXI conversion required. The highest achievable throughputs were mostly limited by these requirements and by the protocols themselves. However, the upside of the internal protocol was that it was more flexible to handle slower read traffic and thus it achieved higher throughputs from slower transactions. All in all, as discussed earlier was the slow traffic mainly caused by external logic and high throughput traffic was choked by the networks themselves but for different reasons.

With other testcases the results showed that the reference network did not have as well-balanced latency dispersion as NoCs had, even though a few separated masters had smaller latencies but the overall average latency of testcases 1-5 were in favor of lower latency versions of NoCs. This can be explained with index-based arbitration where faster traffic caused an additional delay to slower traffic but when there was enough additional delay included, the slower traffic had effect on faster traffic. With the additional test latencies, the overall average latency of the reference network was equally divided for these input-output pairs, but it was approximately two times higher than with the lower latency version of NoC.

Average latencies for tests 1-4 were approximately 1.3 times higher with the reference network but the difference decreased when additional delay was introduced. However, it is important to realize that latency comparison was not fully comparable with testcases 1-3 since a few masters were not transacting within the reference

network which can be seen in Figure 19. Furthermore, simulation results indicated that there was not much latency difference between the lower latency versions of NoCs but rather with the higher latency versions which were in favor of the simpler version. Latency differences were quite equal with lower throughputs, but it had approximately ratio of 1.4 difference when the higher performance was considered.

4.3. Area and power differences

The synthesis results of NoCs were compared with the results from the reference network and the relative comparisons are shown in Figure 20. Total cell area was approximately 2.8 times higher with the simpler low latency version and 3.7 times higher with the high latency version of NoC. Logic area with the sequential and combinational grew almost with the same ratio in both NoCs, although the combinational was a slightly higher. However, a reason why the higher latency version had larger total cell area was due to the transactions which had to be split into four smaller transaction even though the total area grew, there were fewer long data busses to be routed. Besides of logic area, the total power consumption was 2.2 times higher with the lower latency version of simpler NoC and 2.6 times with the higher latency version. It was also observed that the dynamic and static power consumptions were higher with the higher latency versions. In addition, total power consumptions followed the dynamic power consumptions since the dynamic power consumption was approximately 10^9 larger than the static power consumption.

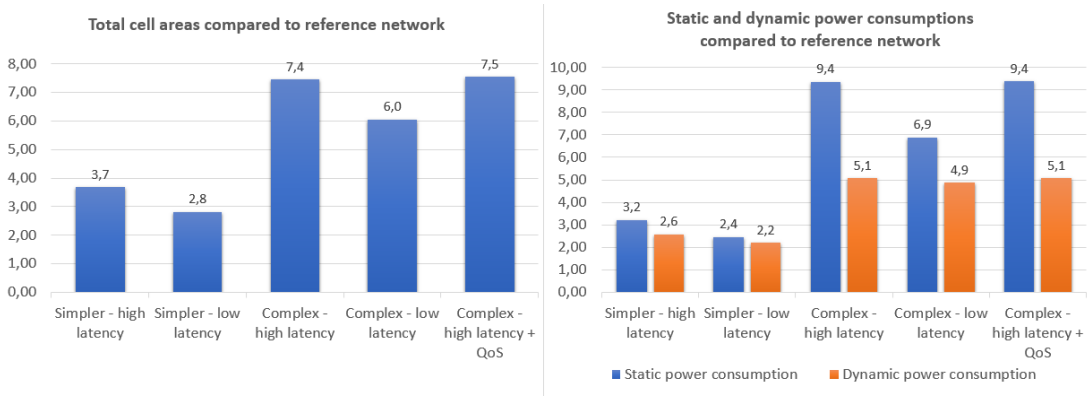

Figure 20. Total cell areas and power consumption compared with the reference network.

Results in Figure 20 with complex versions behaved similarly but the difference between complex versions was larger when compared with simpler versions. For example, it was observed that lower latency version had 6.0 times higher and the higher latency version had 7.4 times higher total cell area than the reference network. Furthermore, the static power consumption was 6.9 times higher with the lower latency version and 9.4 times higher with the higher latency version. However, the dynamic power consumptions of created NoCs were not fully comparable since there was more traffic within the complex versions. Furthermore, the static power consumption with the simpler low latency version grew from 2.4 up to 3.2 with the higher latency version. All in all, results indicated that by including two AHB and two AXI masters and one AXI slave to NoCs, the total cell area increased approximately two times higher and static power consumption three times higher. The dynamic power consumption grew

approximately two times higher with the parallel full speed test. In addition, when the QoS was included in the higher latency version of complex NoC there was 1.4 percent increase in total cell area but the difference with the ratios were lost within roundings.

The used design tool had different optimizing parameters to tune performance which either increased or decreased the total cell area that power consumption followed. For example, when changing lower latencies and wider bus widths into narrow busses and higher latencies, the total cell area increased approximately 30 percent, although with single parameter or small architecture changes it had a few percent differences. However, the logic resources were divided systematically over the network and both masters and slaves used approximately 30 – 50 percentage of resources whereas switches used 10 – 30 percentages. For the other logic, like clock gates, there were a few percent reserved. Figure 21 shows the accurate numbers of resource dispersion over NoCs.

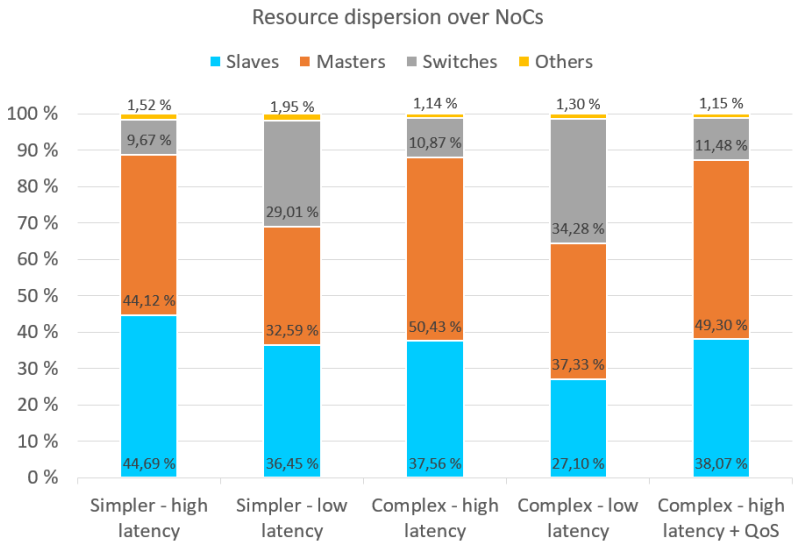

Figure 21. Dispersion of the total cell area over NoCs.

Figure 21 shows that the percentages were divided quite the same way over the different versions but there were differences between simpler and complex versions. The differences can be explained with the numbers of masters and slaves and their protocols. For example, complex networks were included with two AXI masters and one AXI slave. In addition, there were also two AHB masters added but those required less resources than AXI masters. It was also noticed that higher latency NoCs required more logic inside NIs since the wider packets had to be split into smaller packets.

4.4. Comparison of theory, performance estimation and results

Performance estimations from the higher-level testing and theory itself pointed out to be quite accurate corresponding to the simulations results. However, based on Equation 1 there was up to 16 % throughput improvement with the simulation results when no other traffic or external logic were choking the maximum throughput. In general, lower latency versions had more difference with the throughput since even one clock cycle caused a huge difference. In addition, Equation 1 did not take on account of other traffic, parallel transactions from single master and that there was

variation with latency. When averaged simulation results were compared with the results provided by the NoC design tool, the difference was an amount of a few percentages, even though within the design tool it was equally divided over master-slave pairs, but the real test environment caused approximately up to 10 % throughput dispersion with masters 1-5.

The design tool estimated logic area accuracy was within the range of 14-20 % for flip-flops and 20-50 % for combinational when compared with the results from synthesis. Differences were partly due to reason that the tool used a quick RTL-level estimation and that the synthesis was run in the real system and it included optimization. In addition, the design tool estimations were likely somewhat downgraded for the worst-case situation.

Comparison with results and researches was somewhat difficult since there was difference between the used parameters, protocols, connectivity and level of optimization. For example, the used number of switches was lower, frequencies and bus widths were likely different and standards like OCP was not used. For these reasons it is difficult to say whether the frequency was increased with the cost of power consumption or were the bus widths increased with the cost of increased area. Also, different synthesis parameters have a huge effect on achieved results. For example, different temperature corner point influences the power consumption and the used libraries for area. However, the overall glance over results indicated that the lower area and power consumption were achieved.

5. DISCUSSION

The aim of this thesis was to use a NoC design tool for achieving as small as possible area and power consumption without sacrificing the performance under the minimum requirements. On the other hand, it was also important to clarify whether the NoC solution would perform better or worse than the reference multi-layer bus interconnection. For example, it was recognized that the synthesis results of NoC would be likely larger and that performance would have its limitations. Results from different tests, experience gathered and expectations from researches matched quite well to each other. It was expected that NoC solutions were faster to create with design tools and that consumed power and area would be likely larger. Latencies and throughputs were expected to be somewhat higher which would also depend on test patterns, external logic and how NoCs handle its traffic. However, there would still be an additional improvement desired since it was noticed that performance could not be worsened below the certain level as a tradeoff from the larger logic area, but rather that the logic area increased. However, as the back-end point of view it was important to be able to trade wider busses for narrow ones to avoid wiring congestion. All in all, defined objectives were achieved and the findings were useful to identify the use cases where NoC solutions are worth considering over the reference multi-layer bus interconnection.

Based on observations the benefits of NoC-like architecture were somewhat small when networks were homogeneous and performance requirements were low. However, when the performance requirements increased, NoCs did provide more equally balanced and weighted throughputs and latencies whereas the reference network would have required an additional work. Thus, the benefits of NoC began when multiple bus protocols, bus width conversions, different clock and power domains and challenging interconnection and memory mapping were used. For example, it did require more designing time with the reference network to achieve acceptable performance but with NoCs it was only required to enable desired features and optimize through a few iterative loops. However, NoCs were rather plain and small and it was observed that for most of the masters the interconnection itself was not bottlenecking when parallel traffic was concerned. On the other hand, the maximum throughput and minimum latency were defined by the protocol conversions which limited the absolute maximum performance for data paths which were not limited by external logic. Based on these findings and previous research it was suspected that by multiplying the number of interfaces, a NoC solution would be capable of providing much better overall results than the corresponding multi-layer interconnection.

Created NoC solutions were quite fixed to the specifications of multi-layer bus architecture which itself limited the maximum performance at the certain level. For example, throughput could have easily been improved by changing protocols from AHB to AXI, increasing clock frequencies and bus widths and using bursts, even though the achieved minimum latencies it would not have made smaller but rather higher as a tradeoff from the throughput. However, changing bus widths for the internal protocol it would not have made much difference since the external IPs were fixed at certain widths. This leads to the situation where performance requirements must be well known and that IPs need be designed along with the NoC, even though changing frequencies and protocols were relatively fast, it was not possible to fully optimize NoC before the absolute performance requirements and its functionality were defined. In addition, for power simulations it is relevant to notice that only single

power domain was used which could have been split over several smaller domains that would have been shut down when not required. This would have reduced the leakage power consumption.

It is important to realize that NoC solutions are much more than a plain interconnection since those may contain features such as different routing priorities, deadlock avoidance, error handling, virtual channels and adaptive dynamic routing. NoCs may also have several clock and power domains that can be easily implemented with the design tool and are controlled by NoC. With the reference network there would have been additional work required related to clock and power gates and clock domain crossings, for example, when the clock can be gated, and power shut down. In addition, NoC design tools also had means to fix and detect timing violations by easily including buffers and repeaters along the longest data paths which tremendously eases the designing of high-performance networks. Furthermore, commercial design tools tend to provide additional functionality such as CPU and memory interfaces and DMA controllers. However, optimized NoCs had certain functionality and design parameters but those were still considered as a general implementation when compared with the hand-crafted reference network, although the time consumed with designing was significantly higher with the hand-crafted network since the implementation was done separately and all the functionality were verified rather accurately. However, with the NoC design tools, designers can focus on more resources to design different IPs since the basic functionality and architecture are already verified and only the higher RTL-level testing of the whole SoC is left. Another important observation was that the created NoCs were supposed to be used with the top-down design flow which meant that it would have been more beneficial to generate the whole network at once which handles all the traffic. However, such networks might cause system level verification issues since all the submodules cannot be tested separately. There was not any actual problem of creating several smaller networks and to combine them afterwards, but this would limit the overall performance due to the protocol conversions in NIs, have limitations with design flow or have back-end timing issues.

For further testing, it would be interesting to implement a solution which were partly mixed from multi-layer and NoC architectures to avoid conversion latency from certain paths and still be able to communicate with other NIs with increased latency. However, since the tested networks were quite small, a further testing with larger networks would be useful. It would also be interesting to see power consumption results when slow traffic was injected and part of NoC were gated, different power domains were added or how much do the back-end work requirements differ. Testing with different topologies, for example with torus and octagon, would also be interesting. In addition, it would be useful to compare results with multi-layer bus architecture which were also built by a design tool. This would minimize the workload from handcrafted versions, reduce error possibility and give a general view of both cases since the current reference network was quite application specific.

6. SUMMARY

This thesis contained an overall theory about what is needed to take in consideration when network-on-chip architecture is used. In addition, the aim was to use a NoC design tool to achieve an efficient logic area and power consumptions with reasonable performance. It was also desired to gain an overview of NoC solution and observe its performance when compared with the reference multi-layer bus interconnection. Design goals were achieved and the performance of design tool generated NoCs corresponded quite well researches and theory. However, subjects such as NoC topologies available, building blocks, traffic, routing, error handling and error avoidance are discussed. For example, there are several topology choices such as mesh, fat tree, octagon and SPIN and errors such as deadlock and starvation might occur. In addition, NoC building blocks such as NIs and switches are explained. Furthermore, there are several design tools for easing the work flow, such as SonicsStudio® Director, Arteris FlexNoC, ARM CoreLink Network Interconnection and Synopsys DesignWare IP.

This research included one version of reference multi-layer network and five different NoC solutions which were built with a design tool. Two of the NoC solutions were physically smaller and close to the reference network and three of them were more complex versions which had a few master and slave NIs more. For both smaller and complex versions there was one with higher latency and narrower busses and one with wider busses and lower latency. In addition, there was a QoS included for complex NoC with higher latency. Furthermore, several simulations and synthesis were run with attached VIPs and monitors. The synthesis was run with activity files to gain an estimation of the power consumption. For performance it was observed that the reference network was able to achieve higher throughputs when only a single input was used at the time, although with parallel traffic NoC solutions did handle the traffic in more balanced and linear way. Furthermore, with lower throughput injection rate the performance difference decreased. Synthesis results indicated that the reference network had a few times smaller area and power consumption and that resources were divided differently.

A problematic topic of NoCs was that how the traffic was handled. For example, protocol transfer to internal protocol caused a few additional clock cycles which limited its performance. In addition, the tool was meant to be used as top-down which was not the most efficient point of an approach when designing several small networks, as were the tested NoCs. All in all, it would have been more efficient to design larger and more complex networks with high performance requirements rather than uniform and small networks with low performance requirements. Further testing would be required with larger networks but for the smaller ones, NoC solutions might not be the most optimized solution, although the tool significantly saves time and reduces the workload.

7. REFERENCES

[1] Benini L., Giovanni D.M. (2006) Networks on Chips: Technology and tools. Elsevier Morgan Kaufmann, Amsterdam.

[2] Rajsuman R. (2000) System-on-a-Chip: Design and Test. Artech House, Boston.

[3] Architecture Specification: 128-bit Processor Local Bus v4.6 (2004). IBM

[4] User manual: STBus communication system concepts and definitions rev 2 (2012). ST Microelectronics

[5] Specification: AMBA rev 2.0 (1999). ARM

[6] Dimitrakopoulos G., Psarras A., Seitanidis I. (2015) Microarchitecture of Network-on-Chip Routers: A Designer's Perspective. Springer, New York.

[7] Kaushik R., Alakesh K., Abhijit B, Md. Anwar H. (2016) A Multipath Network-on-Chip Topology. In; IEEE International Conference on Information Communication and Embedded Systems, February 25-26, Chennai, India.

[8] Specification: Open Core Protocol 3.0 rev 1.0 (2013) OCP Working Group, https://accellera.org/downloads/standards/ocp/ (04.02.2019)

[9] https://sonicsinc.com (21.1.2019)

[10] http://www.arteris.com/flexnoc (17.7.2018)

[11] https://developer.arm.com/products/system-ip/corelink-interconnect/corelink-network-interconnect-family (17.7.2018)

[12] https://www.synopsys.com/designware-ip/soc-infrastructure-ip/amba.html (18.7.2018)

[13] Specification: AMBA 3 AHB-Lite Protocol V1.0 (2006). ARM

[14] Specification: AMBA 3 APB Protocol v1.0 (2004). ARM

[15] Specification: AMBA 4 AXI and ACE Protocol (2011). ARM

[16] Hautala T. (2016) Performance Analysis of System-Level Bus in A Modem System-On-Chip. Master's Thesis, University of Oulu, Department of Electrical Engineering, Oulu.

[17] Päivänsäde V. (2016) Dynamic Power Estimation with a Hardware Emulation Acquired Switching Activity Model. Master's Thesis, University of Oulu, Department of Electrical Engineering, Oulu.

[18] Bertozzi D., Jalabert A., Srinivasan Murali, Tamhankar R., Stergious S., Benini L., De Micheli G. (2005) NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. In; IEEE Transactions on Parallel and Distributed Systems, January 03, vol. 16, s. 113-129.

[19] Bertozzi D., Benini L. (2004) Xpipes: a network-on-chip architecture for gigascale systems-on-chip. In; IEEE Circuits and Systems Magazine, September 03, vol. 4, s. 18-31.

[20] Rabab Ezz-Eldin, Magdy Ali El-Moursy, Hesham F.A. Hamed (2015) Analysis and Design of Networks-on-Chip Under High Process Variation. Springer, Cham.

[21] Pratiksha G., Shailesh S.C. (2009) Performance evaluation of Network on Chip architectures. In: IEEE International Conference on Emerging Trends in Electronic and Photonic Devices & Systems, December 22-24, Varanasi, India.

[22] Haataja M. (2016) Register-Transfer Level Power Estimation and Reduction Methodologies of Digital System-On-Chip Building Blocks. Master's Thesis, University of Oulu, Department of Electrical Engineering, Oulu. p. 20-29.
[23] https://www.mentor.com/products/fv/questa-verification-platform (19.2.2019)
[24] User Guide: Design Compiler® v. O-2018.06 (2018). Synopsys